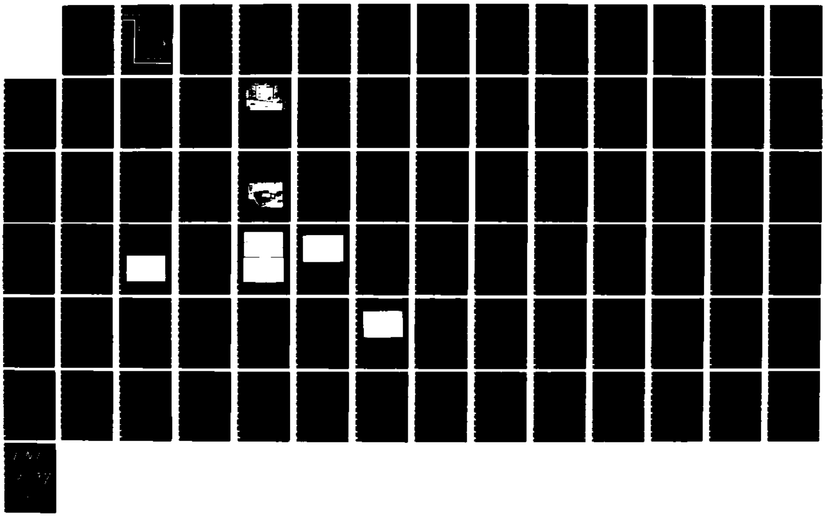
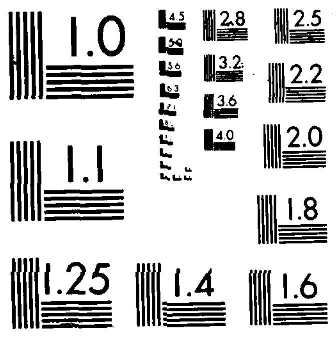
AD-A175 211 ARTIFICIAL INTELLIGENCE TECHNOLOGY FOR THE MAINTAINER'S 1/1
ASSOCIATE(U) DENVER RESEARCH INST CO SOCIAL SYSTEMS
RESEARCH AND EVALUATIO J J RICHARDSON ET AL DEC 86
UNCLASSIFIED AFHRL-TR-86-31 F33615-82-C-0013 F/G 5/8 NL





CROCOPY RESOLUTION TEST CHART

(12)

AFHRL-TR-86-31

AIR FORCE HUMAN RESOURCES LABORATORY

AD-A175 211

ARTIFICIAL INTELLIGENCE TECHNOLOGY FOR THE MAINTAINER'S ASSOCIATE

J. Jeffrey Richardson
Cindy J. Anselme
Kenneth R. Harmon
Robert A. Keller
Bonita L. Moul

Denver Research Institute
Social Systems Research and Evaluation Division
University of Denver
Denver, Colorado 80208

TRAINING SYSTEMS DIVISION
Brooks Air Force Base, Texas 78235-5601

December 1986
Final Report for Period October 1983 to December 1985

AIR FORCE SYSTEMS COMMAND
BROOKS AIR FORCE BASE, TEXAS 78235-5601

TERRESA JACKSON, Capt, USAF
Contract Monitor

HENDRICK W. RUCK, Technical Advisor
Training Systems Division

DENNIS W. JARVI, Colonel, USAF
Commander

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE

ADA175211

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S) AFHRL-TR-86-31
6a. NAME OF PERFORMING ORGANIZATION Denver Research Institute	6b. OFFICE SYMBOL (If applicable)
7a. NAME OF MONITORING ORGANIZATION Training Systems Division	
6c. ADDRESS (City, State, and ZIP Code) University of Denver Denver, Colorado 80208	7b. ADDRESS (City, State, and ZIP Code) Air Force Human Resources Laboratory Brooks Air Force Base, Texas 78235-5601
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Air Force Human Resources Laboratory	8b. OFFICE SYMBOL (If applicable) HQ AFHRL
9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F33615-82-C-0013	
8c. ADDRESS (City, State, and ZIP Code) Brooks Air Force Base, Texas 78235-5601	10. SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT ACCESSION NO.
62205F	1121	09	15

11. TITLE (Include Security Classification)
Artificial Intelligence Technology for the Maintainer's Associate

12. PERSONAL AUTHOR(S)
Richardson, J.J., Anselme, C.J.; Harmon, K.R.; Keller, R.A.; Moul, B.L.

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Final	FROM Oct 83 TO Dec 85	December 1986	92

16. SUPPLEMENTARY NOTATION

17. COSATI CODES

FIELD	GROUP	SUB-GROUP
05	08	
05	10	

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
artificial intelligence, job aiding, troubleshooting, expert systems, knowledge engineering, human-machine interface, maintenance aids

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Shortcomings in the ability of the armed services to maintain sophisticated equipment ~~have long been~~ recognized. Future trends in technological sophistication, personnel resources, and warfare scenarios are expected to aggravate the ~~maintenance~~ situation. In view of these problems and the current Department of Defense policies regarding integrated diagnostics and a reduced reliance on paper-based documentation, the concept of an interactive, portable, computer-based maintainer's associate ~~has been~~ is proposed.

The purpose of this effort was to develop the technology for the Maintainer's Associate based on artificial intelligence techniques and demonstrate a prototype system in the field. The prototype Maintainer's Associate was developed for troubleshooting the F-111 6883 intermediate-level avionics test station. ~~The phases~~ of the project included conceptual design, development and delivery software programming, delivery hardware prototyping, knowledge base development, field demonstration, and analysis of lessons learned.

(Continued)

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT
[✓ UNCLASSIFIED/UNLIMITED] ☐ SAME AS RPT ☐ DTIC USERS

21. ABSTRACT SECURITY CLASSIFICATION

22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE (Include Area Code)	22c. OFFICE SYMBOL
Nancy A. Perrigo, Chief, STINFO Office	(512) 536-3877	AFHRL/TSR

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE
Unclassified

19. (Concluded)

In the course of this work, several ~~important~~ issues were examined: hybrid diagnostics, knowledge engineering costs, user interfaces, and the integration of training and job aiding. The term "hybrid diagnostics" refers to the utilization of multiple sources of knowledge in the development of maintenance expert systems, in particular (a) dependency modeling ~~(potentially~~ derivable from engineering databases) and (b) heuristic expertise of field technicians. In the area of dependency modeling, one source of knowledge identified for the prototype was the test program set of the automatic test station. This information provided the specific measurement values and locations necessary for making measurements during troubleshooting. Knowledge engineering costs were controlled through use of these test program sets and the development of a "glass box" editor which permitted knowledge base modifications during program operation.

The design for the prototype incorporated human-machine interfaces to promote incremental skill acquisition and to mitigate against mental dependence on the Maintainer's Associate. Incremental skill development was also promoted through end-user interfaces which provide a variety of explanations about the reasoning behind the diagnostic process in a given troubleshooting situation.

In a field demonstration, the prototype Maintainer's Associate received highly favorable ratings for ease of use, speed of operation, troubleshooting accuracy, and usefulness for job aiding and training. This project identified additional research issues in using an expert system diagnostic reasoner as the basis for intelligent maintenance training simulations and the need to support mental compilation of test strategies from network dependencies with software tools.

SUMMARY

The concept of a Maintainer's Associate calls for a portable, expert-system-based job aiding and training device to assist inexperienced electronics maintenance technicians. In order to investigate a variety of issues--hybrid diagnosis, knowledge engineering, and user interfaces--a prototype Maintainer's Associate was designed and implemented for troubleshooting portions of the F-111 6883 intermediate-level avionics test station. Both system development software and delivery software are described. In a field demonstration, the prototype system received highly favorable ratings for ease of use, speed of operation, troubleshooting accuracy, and usefulness for job aiding and training. Implications for future development focused on realizing the training potential of the system, enhancing user interfaces, and expanding the problem domain.

ACKNOWLEDGEMENTS

This project was a collaborative effort involving members of five organizations. The Denver Research Institute subcontracted with General Dynamics, Electronics Division (GDE). As subcontractor, GDE provided the Rule-Kit expert system shell and portable hardware units, and worked with Denver Research Institute in their development of the prototype Maintainer's Associate. The authors would like to acknowledge the valuable assistance of John Hinchman and his team members, Mike Morgan, Jim Olsen, and Karen Okagaki.

For their contributions to the knowledge engineering effort, project staff gratefully acknowledge our subject-matter experts, SSgt Susan Schulte and SSgt Thaddeus Thomas from the Special Projects Section, AGE School, 3400 TTW, Lowry Air Force Base, Colorado. Without their expertise, this effort could not have been completed.

The formative field demonstration of the Maintainer's Associate was made possible by the commitment and cooperation of personnel at the 27th Tactical Fighter Wing, Cannon Air Force Base, New Mexico. We would like to thank Col John Wohlsigl and his staff for their comments regarding this work and for providing access to an operational shop. In the avionics intermediate shop, MSgt Andy Wells, TSgt Gary Barrow, and SSgt Theresa Vogt were especially helpful to us in preparing and conducting the field demonstration, while Sgts Barrow and Vogt also provided subject-matter expertise to the knowledge base development.

We would like to thank the contract and technical personnel at the Air Force Human Resources Laboratory, Lt Col Hugh Burns, Capt Terresa Jackson, Brian Dallman, and Gerald Walker for their support and guidance throughout the project.

The authors are pleased to acknowledge Dr. William Johnson and Dr. William Rouse of Search Technology, Inc. as the originators of the term "Maintainer's Associate," which was coined during the 1985 National Academy of Sciences Summer Study on Fault Isolation in Air Force Weapon and Support Systems.

Finally, we wish to acknowledge the contribution of the Social Systems Research and Evaluation Division of the Denver Research Institute. Internal funding provided by the Institute enhanced the contract support of the Air Force Human Resources Laboratory and made possible the completion of the prototype and the preparation of this report.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER I: INTRODUCTION

This report describes an effort conducted by the Denver Research Institute (DRI) for the Air Force Human Resources Laboratory. The focus of this research and development (R&D) effort was the development of the technology for a prototype Maintainer's Associate.

Before this effort is described, the concept of an "associate" for the maintenance technician will be discussed. This concept arose from a variety of problems, policies, and technologies that have merged together to make its realization both desirable and feasible.

Background

Current Maintenance Shortcomings

There are many widely recognized shortcomings with current job aids, training, and technical documentation in electronics maintenance (Richardson, Keller, Maxion, Polson, & DeJong, 1985). Technical documentation, for example, is paper-based and physically bulky. Poor coordination and insufficient cooperation between design engineers and the creators of technical documentation have led to problems of inadequate readability and usefulness; the information is often out of line with technicians' needs and mental approaches to problem-solving. The coordination between technical documentation and instructional materials used for training, as well as between these materials and other resources on the job such as built-in and automatic test equipment, is also insufficient. Keeping paper-based job aids up to date is another problem, because responding to and incorporating suggestions from the field are unrealistically slow. Other current maintenance problems include the need for standardization in the acquisition process, the failings of built-in and automatic test equipment, the demand for more skilled technicians from a less skilled recruit pool, and logistical problems throughout maintenance information support systems (Richardson et al., 1985).

In addition to these shortcomings, there are a number of trends which compound today's maintenance task and threaten to make the future of supporting weapon systems even more difficult. First, advances in technology have complicated rather than simplified maintenance because technology tends to increase functionality but not reliability. Second, personnel resources are diminishing. Highly skilled people are needed [illegible] military at a time when the supply of young persons of all aptitudes is declining and competition with industry for experienced technicians is great. The services cannot rely on counteracting advancing technology's impact on maintenance by recruiting more and brighter personnel. A third trend concerns the operational requirements of the future. Battle scenarios for the late 1990s and early 21st century call for the ability to sustain intense surges; the need for [illegible] and the capacity to mobilize against a more capable [illegible] Resources Laboratory, 1984). All of these requirements [illegible] maintenance.

In addition to the current maintenance situation, there are two other factors that have contributed to the concept of a maintainer's associate: (a) Department of Defense (DoD) policy, and (b) technological advances.

DoD Policy Initiatives

Foremost among policies that have contributed to the associate concept is that which pertains to Integrated Diagnostics. This policy states that all life-cycle concerns relevant to maintenance should be considered in an integrated fashion. Integrated Diagnostics is a structured process which maximizes the effectiveness of diagnostics by integrating pertinent elements such as testability, automatic and manual testing, training, maintenance aiding, and technical information. The goal is to minimize equipment failures by addressing maintenance and logistics support problems at the beginning of the design phase of a system (National Security Industrial Association, 1983, 1984b).

Another important policy was the DoD logistics R&D initiative to replace paper technical orders with an interactive maintenance aiding device (National Security Industrial Association, 1984a). There are numerous ongoing R&D programs working toward this goal. In the Air Force, the Integrated Maintenance Information System (IMIS) program (Johnson, 1981) is the first program to clearly define the functionalities of a system to support maintenance technicians' information needs through electronic means. In the Navy, similar programs are the Personalized Electronic Aid for Maintenance and the Integrated Diagnostics Support System. The phrase "maintainer's associate" was first used in the 1985 National Academy of Sciences Summer Study on Fault Isolation in Air Force Weapon and Support Systems (National Academy Press, in press) to describe such a device. One of the recommendations that emerged from this effort was that the Air Force should immediately structure a program to develop a maintainer's associate system for a specific application in the near future.

Technological Advances

The concept of an interactive job performance aiding and training device is feasible because of recent advances in artificial intelligence (AI) techniques applicable to physical systems, especially in the area of computer programs called "expert systems." Expert systems are able to explain their reasoning, deal with uncertainty, and expand to augment their competency. Although there are other AI applications to maintenance, such as design for testability and maintainability, embedded test ("smart" built-in test), off-line test (automatic test program generation), and logistics decision support, expert systems can be used to address the human resources problems of developing and supporting skilled technical personnel through the concept of a maintainer's associate.

Development of a Maintainer's Associate

Due to the fact that R&D in the area of a maintainer's associate is still in the exploratory development stage, there is sometimes confusion between the concept of an associate, the design or plan for an associate, and the actual prototype device that has been developed and demonstrated. To avoid this confusion, the authors of this report will refer to the concept as it is discussed in the following sections; i.e., in terms of the idealized scenario which describes how a human technician and a portable machine should act in cooperation to troubleshoot maintenance problems. The design or plan for the associate is the overall scheme for the prototype which will eventually include features which are attainable goals, but not all of which were realized in the present effort. The name "Maintainer's Associate" will, throughout this report, refer to the actual prototype device itself and its features as developed and demonstrated by DRI.

System Concept

The concept of an associate system involves a computer-based device with three basic functions: an electronic information resource, a job aid, and a trainer. Figure 1 illustrates how the system is expected to function as a flexible integrator of different information resources. The system's internal memory should contain a file of engineering design drawings, schematics, illustrated parts breakdowns, and basic theories of operation. The system should also interface the global data supporting the aircraft under repair. The fleet's maintenance history and an aircraft's onboard diagnostic and operational data systems could be accessed by data links and made available to the technician through the associate. In addition to drawing on this maintenance corporate memory for the aircraft, the associate should also build the memory by acting as an interface to a management information system (MIS). The technician would use the associate to file reports on work in progress and to record new insights of potential use to the technician's peers and successors.

As a job aid, the associate concept involves an expert system capable of providing advice and direction for performing diagnostic tasks. The system should enable unskilled technicians to perform as if they were skilled and to work cooperatively with skilled technicians to solve difficult diagnostic problems and capture these insights for subsequent use by less experienced technicians. These system-user interfaces are expected to elevate the associate above the traditional, "cookbook" job performance aids. They should promote, monitor, and use the skill of technicians in a cooperative machine/human system. In contrast to current automatic test equipment technology where a fixed sequence of tests is followed, the associate is planned to operate from an expert system knowledge base that can develop sequence of tests "on the fly." Thus, the associate concept permits the technician to peruse interactively the space of possible solutions to a problem. The technician would be able to observe the supporting data on which the expert system has based its "solution" and to work in conjunction with the computer by assessing its conclusions.

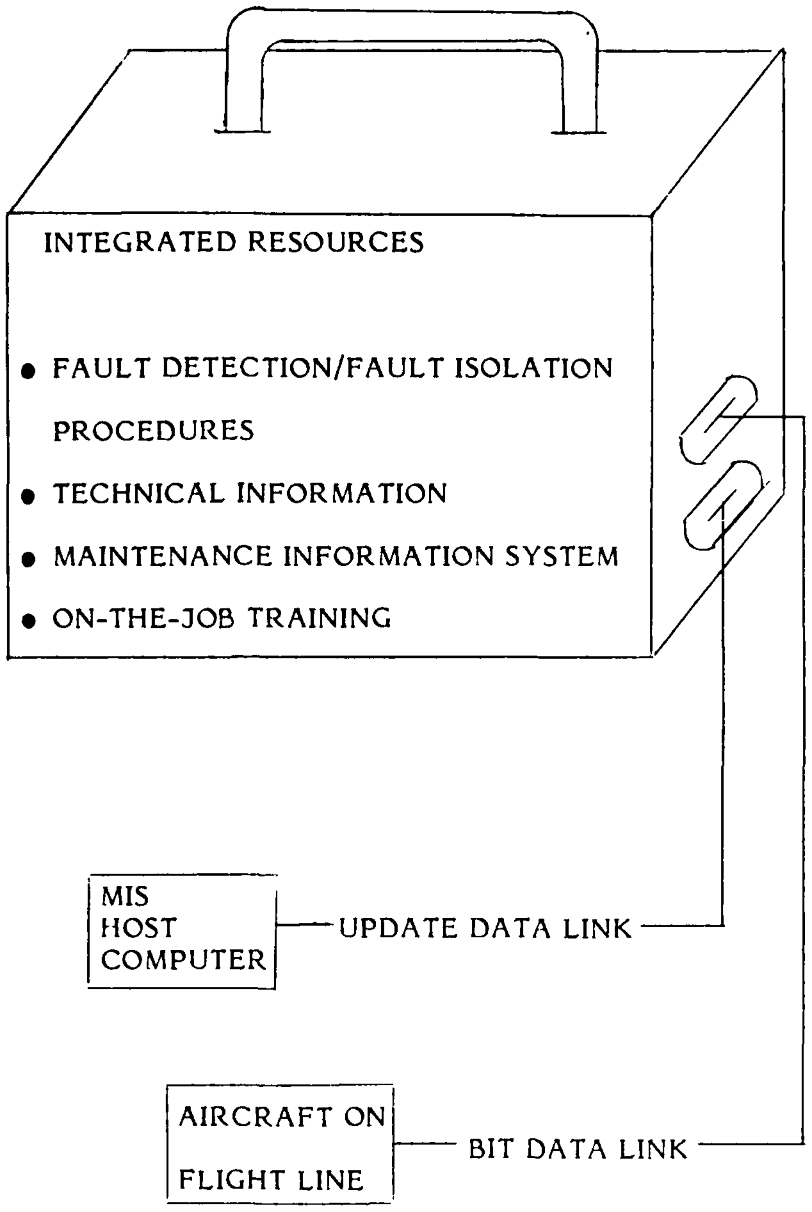


Figure 1. Illustration of the Maintainer's Associate System Concept.

As a trainer, the associate should be able to work with the technician whose skill is still developing. Operating in a tutorial mode, the associate should provide troubleshooting practice and track the technician's progress. The technician would be encouraged to anticipate the expert system's diagnostic reasoning. Given its foundation in AI technology, the associate should be able to justify its decision to its human colleague. The technician should be able to begin a tutorial session at convenient times, either during downtime in regular maintenance activities or during time specifically set aside for training. Since a knowledge base is the source of diagnostic expertise, the associate should be able to respond to the technician at the appropriate skill level, based on a model of the user and instructional principles. The model may also be tied to the technician's personnel record and, in the aggregate, to the records of the entire maintenance labor force. Full realization of the maintainer's associate concept would integrate the traditionally separate concerns of training and job performance aiding.

System Benefits

Several benefits are expected to result from the successful deployment of a maintainer's associate in the field. As digital information processing replaces the growing volume of paper technical documentation, the cumbersome bulk of paper aids ceases to be a problem. In the digital medium, information is accessed faster and manipulated more easily. Another projected benefit of the associate is the promotion of technician excellence, because the system is intended to act as a skill multiplier for novice technicians and as a skill integrator for skilled technicians which would capture the corporate memory of a maintenance corps regardless of personnel changes.

The risks involved in developing an associate within a 5-year time frame are manageable. This statement is supported by three observations. First, the AI technology upon which an associate is based has been developing through R&D for over a decade. Second, demonstration prototypes have been developed for nontrivial systems. Third, the AI software needed for an associate has appeared in the private sector, indicating that the risk has been reviewed and deemed worthwhile by those with substantial economic interests. The level of resources needed to develop an associate for deployment with a weapon system is likely to be commensurate with the data costs of the weapon system acquisition. According to a 1983 Armed Forces Comptroller report on weapon system life-cycle cost, 5% of the acquisition cost for a weapon system is for data (Lahore, 1984). However, the "know-how" developed in first efforts will be amortized across the succeeding applications.

Target Environment and Task

The target environment for the prototype Maintainer's Associate was the intermediate-level avionics repair shop for USAF F-111s. Figure 2 shows the F-111 6883 Converter/Flight Control Test Station which is used to fault-isolate malfunctioning line replaceable units ("black boxes") previously removed from aircraft on the flight line.

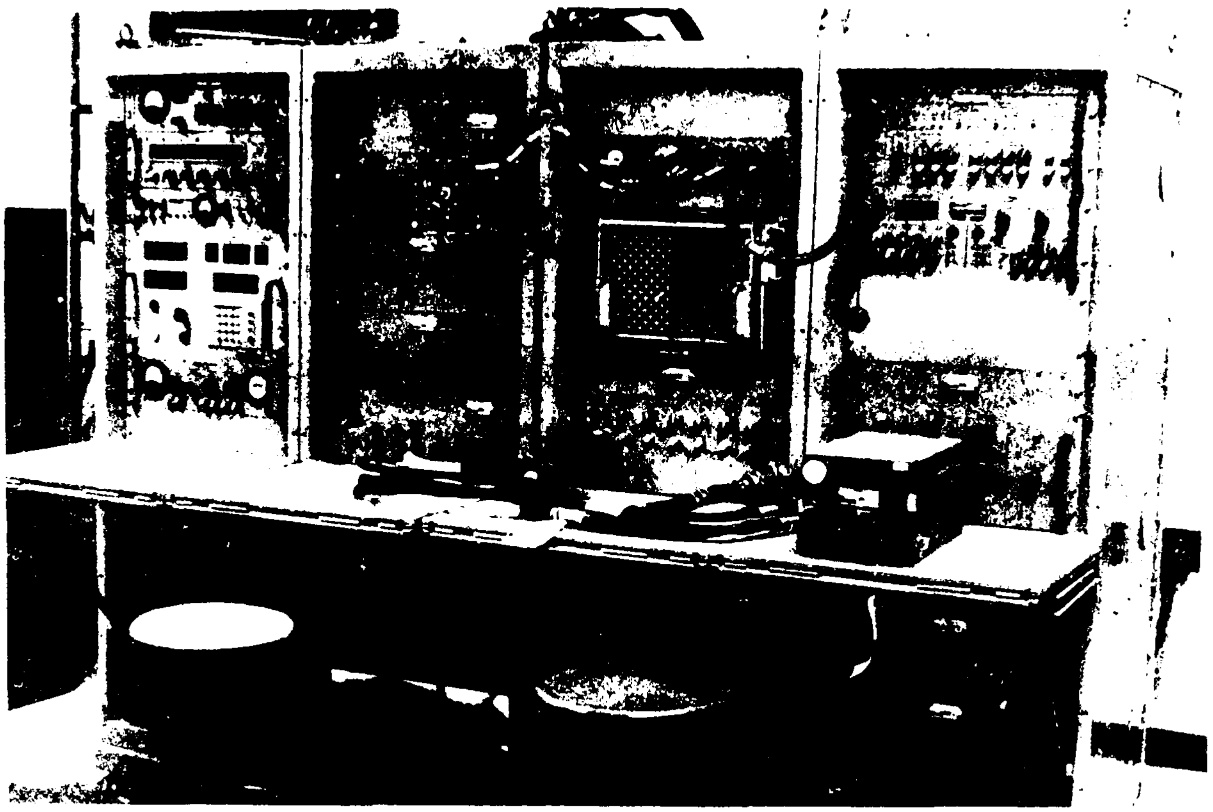


Figure 2. F-111 6883 Converter Flight Control Test Station.

Automatic test stations such as the 6883 were originally introduced to reduce or eliminate the need for manual troubleshooting. However, manual troubleshooting is still required to isolate faults which the test station cannot find within a unit under test (UUT) and to isolate faults within the test station itself. Although test stations are provided with a self-test capability, most technicians prefer to troubleshoot them manually.

Figure 3 provides a simplified diagram of how an automatic test station works by switching stimulus signals through a patch panel and adapter (test station interface) to the UUT. Response signals from the UUT flow back through the adapter and are switched to measurement devices which compare the received signal to an expected signal. This signal path is termed the "test loop," and there is one test loop for each and every test applied to the UUT. The UUT selected as the application testbed for the prototype Maintainer's Associate was the Feel and Trim Computer, which is tested by over 400 tests.

If the test station malfunctions, this is manifest during a specific test. The station would indicate a certain malfunction in the UUT which, when repaired, stills checks out "bad." The key to troubleshooting the test station is that the probable causes of the test failure are limited to those components along the currently active test loop. This is why technicians prefer to troubleshoot the test station manually. They can use the current test information to narrow the search for a malfunction, whereas the test station's self-test sequentially checks all components. The maintainer's associate design was developed to use this same test loop strategy that experienced technicians use to troubleshoot the test station.

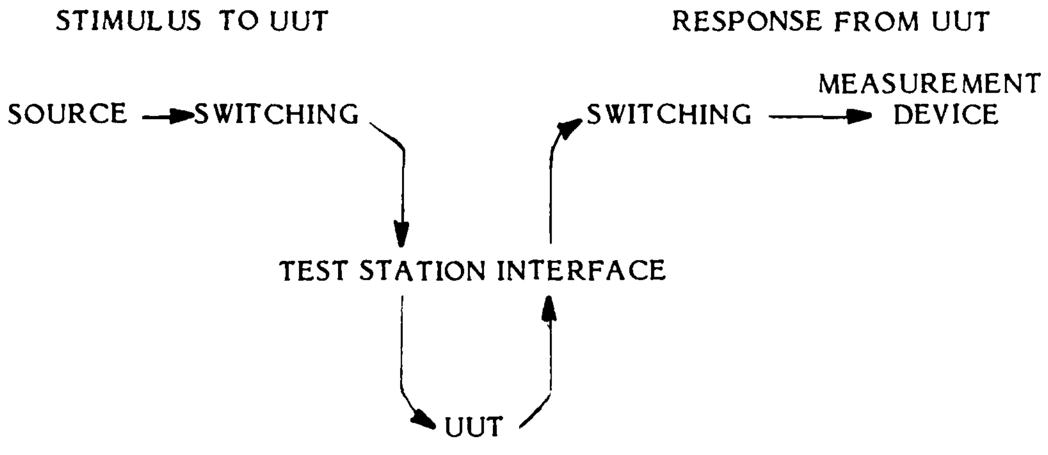


Figure 3. A Simplified Test Loop for Automatic Test.

Because the purpose of this R&D was to investigate intelligent maintenance technology rather than build a system for field use, the diagnostic coverage of the knowledge base was limited. The utility of a prototype would be demonstrated if the system could fault-isolate from the test station as a whole to the next level of repair; that is, to one of the 29 test station replaceable units (TRUs). Achieving this goal would illustrate the fault isolation process through one level of refinement. In order to demonstrate a second level of refinement, a specific TRU was chosen for further fault isolation. (For field use, an associate would, of course, continue refinement within all 29 TRUs until the appropriate level of repair was reached.)

The diagnostic coverage of the prototype was also limited to troubleshooting the test station when the UUT was the Feel and Trim line replaceable unit (LRU). This LRU represents about 50% of the test station work-load. Because the test strategy depended on troubleshooting the test loop, the approach used was context-sensitive to the particular unit under test and its associated test program set and set of test loops.

Goals and Objectives

The overall objectives of this effort were to conduct exploratory research concerning the role of an associate and technologies for further development, and to develop and demonstrate a prototype Maintainer's Associate.

Exploratory Research

The heart of maintenance is troubleshooting. Thus, a thorough understanding of diagnostic problem-solving was a prerequisite to understanding and designing the prototype Maintainer's Associate. Further, because the overall goal of this effort was the development of an interactive maintenance system, both technician and expert system perspectives on troubleshooting were investigated. Human troubleshooting, and its implications for intelligent maintenance aids, was reviewed in a previous report (Keller, 1985). The present report focuses on expert system approaches and specifically on a knowledge acquisition strategy called "hybrid diagnosis" which uses knowledge about the structure of the system under test, as well as knowledge about fault/symptom associations.

Diagnosis is a special kind of problem-solving called "classification problem-solving" (Clancey, 1984), in which the problem-solver selects from a set of pre-enumerated solutions. Diagnostic test strategies are either precomputed, as in the traditional automatic test equipment approach to diagnostic test; or they are developed in real time as a diagnostic session proceeds, as is typical in the AI approach. In either case, the set of "right answers" (i.e., the potential faults) toward which a successful strategy converges, is known in advance.

The key to classification problem-solving is hypothesis refinement (also termed "establish-refine"). A fault is isolated to one of a set of probable causes at a given level of abstraction ("established"); then, the probable cause is broken down into more finely detailed probable causes ("refined"). This process is repeated until the fault is isolated to a sufficiently small probable cause set (Chandrasekaran, 1983; Tanner & Bylander, 1984). This strategy is similar to the three-level military maintenance philosophy of field, intermediate, and depot maintenance. However, even when it is applied within one maintenance level, this strategy of "divide and conquer" has diagnostic power and efficiency.

The refine step of the establish-refine strategy calls for selecting the one correct item from a set of possible items. For troubleshooting, the refinement process itself consists of five steps which, when repeated iteratively, converge on a fault at a given level of abstraction. These five steps are: (a) decide whether further diagnostic refinement is warranted; (b) select where to make the next observation based on maximizing the expected information gain per unit cost; (c) identify the expected value at the selected observation point; (d) make the observation; and (e) determine the implications of this observation in terms of component blame or innocence. This process may be summarized as a cycle of making observations and computing entailments (de Kleer, 1984). There are two ways of implementing this five-step refinement process: the specification-based or symptom-based approach.

Specification-based diagnosis. The specification-based approach, often termed "deep reasoning" (also causal, topographic, topologic, or state-based reasoning), solves diagnostic problems by reasoning from a device model (Genesereth, 1984). A symbolic representation of the components that constitute a device, together with their input/output behavior and interconnections, enables reasoning directly from a "deep theory" consisting of information about intended structure and behavior. Figure 4 shows the advantage of the specification-based approach. The basic representation, the device model, is not specialized for any specific task, such as diagnosis, and therefore can be used for multiple purposes in the design and support of weapon systems. A related but simplified form of specification-based diagnosis is logic modeling, in which connectivity is modeled but not module input/output behavior.

In the specification-based approach, knowledge is represented as propositions that are simple statements known to be true. Examples of such statements are "the output of the signal generator is connected to the input of the oscilloscope" or "the amplifier is bad." Through the use of resolution-based theorem proving (Genesereth, 1984), or other techniques (Davis, 1984), these statements are combined to develop new propositions. Lists of suspected faults and tests to be made will have certain forms when represented propositionally. The basic idea is to derive these forms from the current set of propositions when a list of suspects or a measurement is needed.

Using only the device model, the composite behavior of the system can be derived by propagating individual component behavior through the connectivity network (Davis, 1984; de Kleer 1976; Sussman & Steele, 1980). Knowledge about this behavior is also constrained by applicable network laws, such as Ohm's and Kirchoff's laws.

With the specification-based approach, the device model of the system under test is in the engineer's mind if the diagnostic program is being developed directly by a test engineer. If the diagnostic program is AI-based, then the device model is in a computer. In either case, this model is used to generate expectations about circuit measurements, which are compared with actual measurements. Discrepancies between expected and observed values are then used to rule out certain components and cast suspicion on others. As described in the five steps of the establish-refine cycle, the new state of the model is used to select the next measurement, based on the maximum information gain.

Symptom-based diagnosis. The symptom-based approach, often termed "shallow reasoning" (also pattern matching, evidential, associationistic, or empirical reasoning), solves diagnostic problems by manipulating a set of associations between symptoms and faults. With this approach, the associations between symptoms and faults represent a compiled form of knowledge which is streamlined and conditioned for the diagnostic task. The principles and models from which this knowledge is derived are not always readily accessible to the problem solver or may even be unknown or forgotten. Often symptom-based knowledge is heuristic in nature (i.e., fallible) and is based on experience more than reasoned causal derivation.

Generally, the associations in the symptom-based approach are founded on simple empirical observations, but they may also be logical consequences deduced from the device model of the system under test. As such, these systems represent diagnostic knowledge in a compiled form. Here, the device model and general diagnostic algorithm are used to compute a special-purpose data structure tailored to the diagnostic task.

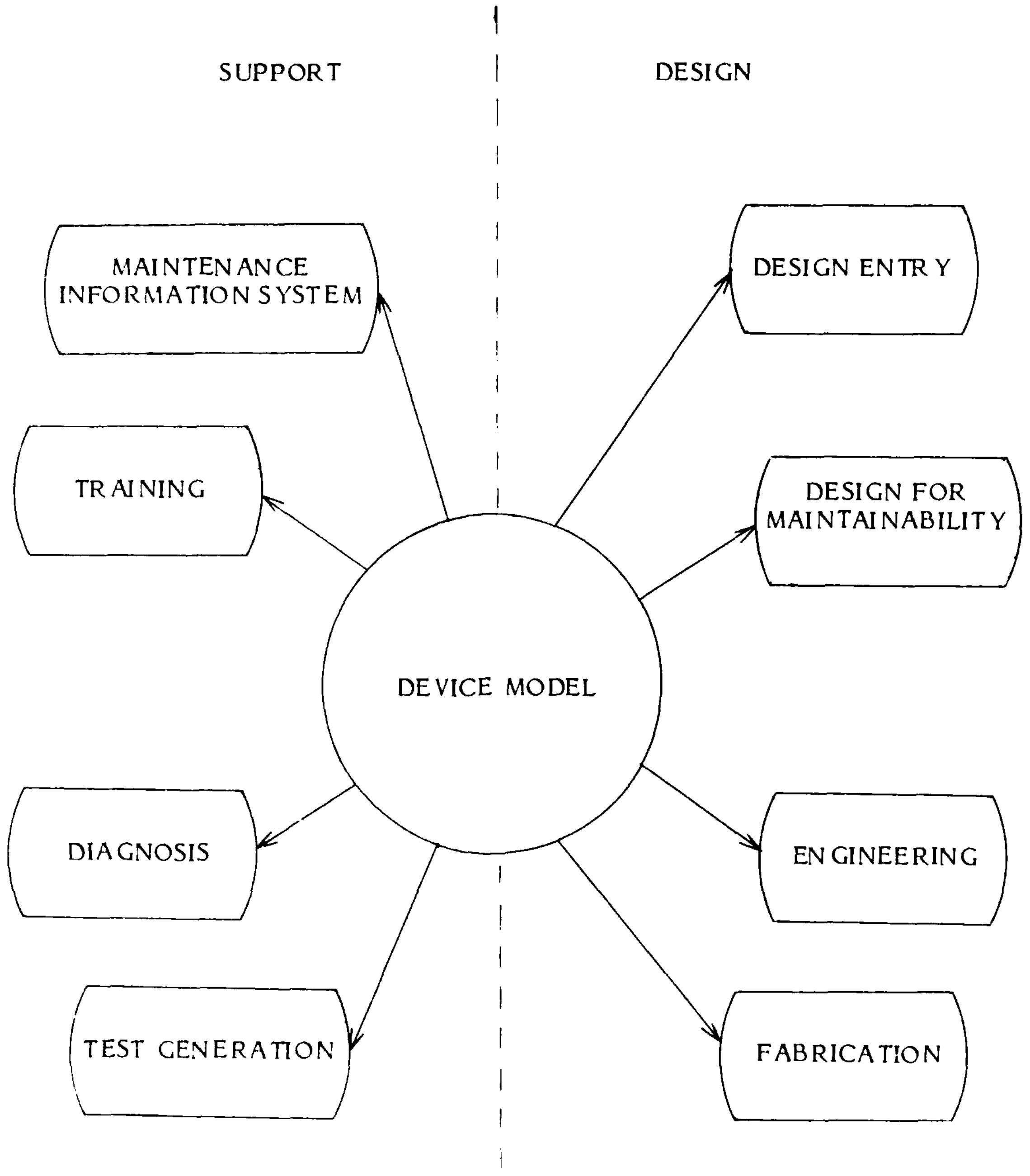


Figure 4. Illustration of How a Device Model Relates Design Features and User Functions.

Hybrid diagnostic reasoning. AI systems have been developed for both the specification-based and symptom-based approaches. Human problem-solving technicians also use either approach, but generally prefer to use shallow reasoning when possible and resort to deep reasoning only when forced to do so (Rouse, 1984). AI systems can employ a similar strategy of using both techniques as needed but to date they do not, tending instead to be one or the other, but not hybrid combinations. The two approaches, however, are inherently interrelated. For example, there must be a causal explanation for every empirical fact. The specification-based approach focuses on the causal explanation; the symptom-based, on the known fact. With one exception, described by Fink, Lusth, and Duran (1984), expert systems that capitalize on the potential synergism between the two approaches do not exist.

Table 1 provides a sample of literature relevant to computer-based diagnosis. An in-depth review of work on specification-based diagnostic reasoning is found in King (1982), and one volume of the journal Artificial Intelligence (Bobrow & Hayes, 1984) is devoted to qualitative reasoning about physical systems, bringing together research previously published in scattered conference proceedings. Artificial Intelligence in Maintenance (Air Force Human Resources Laboratory, 1984) contains a number of the works cited in the table.

Prototype Design and Development

Specific objectives for the development and demonstration of the prototype included: (a) demonstrating a Maintainer's Associate that serves as a skill multiplier for inexperienced technicians and as a skill integrator that uses and captures the corporate memory of skilled technicians, (b) developing an efficient authoring system for developing the Maintainer's Associate knowledge base, (c) constructing a portable Maintainer's Associate hardware unit for the end-user, and (d) collecting and analyzing responses from members of a maintenance organization for use in guiding future efforts.

Chapter 2 discusses design detail, and the implementation of the design is discussed in Chapter 3. Chapter 4 describes the knowledge engineering effort conducted for the prototype. The results of the system demonstration efforts are presented in Chapter 5. Chapter 6 provides conclusions and implications for the entire effort.

Table 1. A Sample of Literature Relevant to Computer-Based Device Diagnosis

Diagnostic Approach	Literature Reference	System Name
Logic Modeling	Wong and Andre (1976, 1981)	---
	Andre and Wong (1975)	---
	Longendorfer (1981)	---
	Cramer et al. (1982)	---
	DETEX Systems, Inc. (n.d.)	LOGMOD
	Simpson and Balaban (1982); Simpson and Agre (1983)	STAMP
	Cantone (1984); Cantone et al. (1983, 1984)	INATE
Specification-based	Brown and Sussman (1974)	LOCAL
	Stallman and Sussman (1977)	EL
	McDermott (1976)	DESI
	Brown (1977)	WATSON
	Brown, Burton, and de Kleer (1982)	SOPHIE
	Genesereth (1982)	DART
	Davis (1983); Davis et al. (1982); Hamscher and Davis (1984)	---
	Pipitone (1984)	---
Symptom-based	McDermott and Brooks (1982)	ARBY
	Hinchman and Morgan (1984); Williams and Hinchman (1983)	IMA
	Bonissone and Johnson (1984)	DELTA
	Davison (1984)	---
	Laffey, Perkins, and Nguyen (1984)	LES

CHAPTER 2: DESIGN OF THE MAINTAINER'S ASSOCIATE

Three general design issues that were encountered and resolved in the design of a maintainer's associate are discussed in this chapter. First, the issue of hybrid diagnosis is considered, which integrates both specification- and symptom-based knowledge in the same knowledge base and interprets both with a single inference engine or reasoning strategy. Second, the target equipment presented a special design challenge because the automatic test equipment is a reconfigurable system; that is, its device model is not static but varies for each of over 400 different test number states. The problem of designing for reconfigurable systems is therefore examined. Third, design issues related to the user interfaces that support system authoring, skill multiplier, and skill integrator concepts are discussed.

Hybrid Diagnosis

For each level of refinement in the hierarchical decomposition approach to problem-solving, the compiled test tree spans from a parent node (a device module at one level of the hierarchy) to a set of constituent component nodes (that module at the next level of refinement). The test nodes between these two levels correspond to subsets of constituent components. The diagnostic test tree shown in Figure 5 represents the compiled knowledge of a specification-based approach to diagnosis which is now in a form compatible with symptom-based diagnosis. The tree is essentially deterministic in character; given various outcomes of tests beginning at the root node of this tree, the problem will resolve to the correct faulty subcomponent at the next level of refinement.

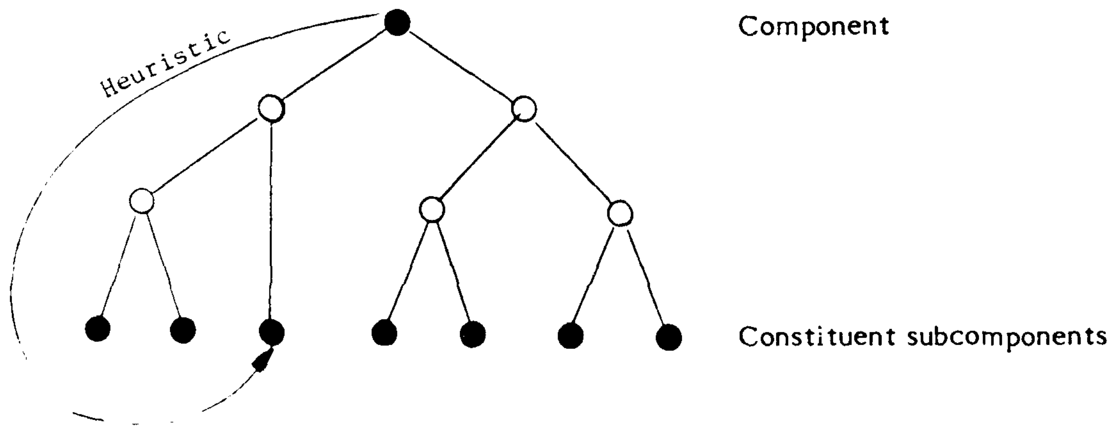


Figure 5. A Specification-Based Test Tree with an Overlaid Heuristic Inference.

The strength of the symptom-based approach to diagnosis is in the use of heuristics. These are "rules-of-thumb" which capture knowledge derived from experience. Although these rules are device-dependent, they often have a great deal of diagnostic precision that is not derivable from a structure model. The rules of inference in the symptom-based approach to diagnosis are in the form "symptom implies fault." Because there are few or no limits on what can be described as a symptom, the rules can capture quite complex patterns that serve as signatures to specific faults. Often these heuristic rules can shortcut several levels of diagnostic tests generated by a specification-based approach. Heuristic inferences of this sort can be represented by an arrow indicating that, given a certain symptom pattern, a particular subcomponent is directly suspected to be at fault as shown in Figure 5. It is sometimes necessary to backtrack and undo diagnostic inferences based on heuristic rules, because a heuristic is not infallible. When this happens, control moves up in the diagnostic tree instead of down, and the previous path that did not yield a solution is ruled out from further consideration.

In the design for the prototype, specification- and symptom-based diagnostic approaches were integrated by compiling the specification-based information into a diagnostic decision tree upon which symptom-based heuristic rules were overlaid. This integration capitalized on the ease of developing a diagnostic knowledge base that was characteristic of the specification-based approach, while at the same time incorporating heuristic knowledge in the form of symptom/fault associations. Two important objectives of this design effort to use hybrid diagnosis included documenting how knowledge engineers build a diagnostic tree so that the process can be computer-aided or computer-automated and determining the relative proportions of specification- and symptom-based knowledge used in diagnosis. The results of these two activities in prototype development are discussed in Chapter 4.

Reconfigurable Systems

In designing and developing the knowledge base, the following questions had to be addressed: Since the test station can be in any of over 400 states (depending on the test number it is executing), would there need to be one set of rules in the knowledge base for each of these states? Further, what is the impact of the state of the system under test on the structure of the diagnostic knowledge base?

These questions were addressed by realizing that whatever changes in state the test station goes through, the test loop remains invariant at an appropriate level of abstraction. In other words, for any current state of the station, a signal is routed through the UUT to a measurement device, as previously described in Figure 3. Thus, at the first level of refinement, it is possible to view the test station as a number of generic regions along the path of the abstract test loop. For each test, the test station is sent a sequence of programming instructions which set the conditions required to perform the given test. If it is assumed that test station failure is always associated with a specific test number, it is then possible to determine the specifics of the signal path and the expected signal values at the various points along the test loop. Given this perspective, only one generic diagnostic test tree must be developed.

This diagnostic test tree is a hierarchy of tests which splits the set of all probable causes of failure (represented by the root of the tree) into small subsets until a failure can be isolated at the current level of refinement. Developing this tree requires deciding where topologically to measure and the consequences of a measurement in terms of absolving or blaming components. For any specific test, these requirements translate into knowing the precise physical location for test and the correct signal value to expect.

There are several alternatives for deriving these expected measurement parameters. One approach is to interface the expert system with a correctly functioning piece of hardware. This is the approach taken in signature analysis. A second approach is to query a standard circuit simulator, as is done in SOPHIE I (Brown, Burton, & de Kleer, 1982) or STEAMER (Hollan, Stevens, & Williams, 1980). As a third approach, a device model may be used, with expected measurement values computed through constraint propagation and dependency-directed backtracking (Davis, 1984; Genesereth, 1984; Sussman & Steele, 1980). A fourth alternative is to have subject-matter experts or knowledge engineers develop expected values mentally and enter these values into the expert system as data, as was done in Pipitone (1984).

In the present effort, a fifth alternative was employed. A file of expected measurements was developed from the test program set for the automatic test station and the tabular and schematic information available from the technical documentation. This file of expected measurements was generated by a special computer program called a parser. The parser was designed as an editor so that it could be used with other test stations or with other equipment with sets of data organized by system state stipulating expected signal values and locations. Details of the parser are discussed in Chapter 3.

User Interfaces

In order to implement the desired functions of the maintainer's associate concept, DRI designed a series of user interfaces. These interfaces enable the technician to be both system-builder and end-user, because both are important to the successful development, use, and maintenance of the database on which the system operates. In the following sections, the design specifications for three optimal user interfaces are presented: an authoring system, skill and [illegible] interfaces, and skill integrator interfaces.

The Authoring System

The demonstration of tools for developing the maintainer's associate is nearly as important as the demonstration of the prototype itself. Large-scale implementation of these devices would be impossible without the [illegible] to efficiently develop, debug, and maintain their knowledge bases. As it was expected that some knowledge base debugging and maintenance would be conducted during system operation, it was vital that the authoring tools be integrated with run-time software so that a knowledge engineer or author can transfer effortlessly between using the device and editing its knowledge base. To enhance this process, various types of information must be visible and accessible to the user. The editor was therefore designed to augment domain-specific messages with all other pertinent information regarding the state of the expert system architecture. The visibility of this information suggested that the editor be termed a "glass box."

Two procedures were designed to implement changes in the state of the system. The first method was single-stepping, in which states shift step-by-step in accordance with the expert system's inference engine. The second approach was interaction-stepping, where the system state is visible as the system pauses for user interaction. Because this design allows the system author to step the expert system through its algorithm, viewing the resulting states along the way, the editor is also termed a "runnable editor."

The Skill Multiplier Interface

At the minimum, a maintainer's associate must prompt the user for only necessary information and inform the user of the eventual diagnosis. In this mode, the system would operate as a fully proceduralized job performance aid (FPJPA). However, traditional FPJPAs neither promote active learning nor recognize any differences in user competence. In this section, a number of potential skill multiplier interfaces which were designed to support on-the-job learning are described.

The "how-to" interface. The purpose of this interface is to augment the normal interaction message; typically it is a multiple-choice question regarding a specific signal or test, with more detailed information about where to locate the signal or how to perform the test. For example, the interaction frame might ask the technician to use a digital voltmeter and report the value. If the technician does not know how to do this, the how-to interface would provide details. The level of detail could be structured hierarchically so that the user gets just the right amount of help. Displays to be provided in this interface, as in interactions themselves, combine text and graphics.

The "where-from" interface. This skill multiplier interface involves the diagnostic process itself rather than the details of physical manipulations. The where-from function answers the question: "What has happened so far?" Lists of previously executed interactions and their answers, assertions in working memory, and probable causes would be provided by level of refinement. Also, if evidence included a special rationale entered by the knowledge engineer during knowledge base development, this stored explanation would be accessed through this interface.

The "where-to" interface. This interface also relates to the diagnostic process and answers the question: "Why are you asking me that question now?" or "Why should I conduct this test?" In response to a where-to query, the system would explain what evidence may be obtained by conducting this test and how that evidence may help discriminate among current probable causes. This information would be presented as an English-like rendition of the rules of evidence which caused the interactions to queue up. The user would also be able to request to see the other tests which queued up for this level of refinement and their associated evidence or any canned messages associated with the evidence.

These skill multiplier interfaces would allow the user to obtain more information about the ongoing diagnostic session. This information is accessed, as needed, under the user's control and thereby promotes skill development on the job. This should prove effective because the user is presented with this information only when requested and always in context.

The references interface. The expected values of various measurements are provided to the technician by the system through message interactions. In developing the database of expected values, indices to the sources of information can also be saved. Through this interface, it would be possible for the technician to access this additional technical documentation. In answer to a technician's question (such as "How did you know to check pin XYZ?"), the system would direct the user to the appropriate reference for that information.

Future implementations of a maintainer's associate could extrapolate this interface to a general context-dependent index to all technical information about the system under test: theory of operation, setup, checkout, calibration and alignment procedures, schematics, tables, illustrated parts breakdowns, and removal and replacement procedures. Having this information stored on-line as a relational data base alleviates the two principal shortcomings of current documentation: the physical bulk of paper-based documentation and the difficulty in finding and cross-referencing needed information.

The tutor interface--maintenance troubleshooting simulation. The four skill multiplier interfaces described above were designed to be available to the user during a consultation at any point in the current diagnostic process. In contrast, the tutor interface would be a distinct, special-purpose mode of operation that could be selected while the user has some spare time or during a time period allocated to formal study. In the tutor mode, the basic consultation process would be reversed: Instead of the associate fault-isolating for the user, the user would fault-isolate for the associate. Rather than providing input requested by the maintainer's associate, the user learns to lead the associate by generating the diagnostic steps that it would follow. A strategy would have to be developed to avoid potential natural language problems. For example, the tutor might display a list of probable causes, including one that does not belong, and ask the user to identify the distractor. Similar means of forcing the user to anticipate the associate's processing would be developed for the other steps in the establish-refine cycle. The exact sequence the user must follow, given a selected fault for maintenance simulation, would be generated by following the path that leads from the fault back up to the root of the diagnostic tree. The tutor would build this path bottom-up, and then force the user to follow it top-down.

In future implementations, this interface could be linked with the status records of the technician's on-the-job training curriculum. If the objective of the curriculum was to enable the technician to troubleshoot any fault known to occur, the diagnostic tree itself would handily represent a hierarchical description of the curriculum; i.e., the technician's competence could be modeled as an overlay on the diagnostic tree with the portions that the technician has mastered marked as such. Then, employing a suitable sequencing strategy, a new tutorial simulation exercise could be selected in accordance with both the training curriculum and the trainee's demonstrated competency.

Through the tutor interface, the technician should learn the basic establish-refine approach to diagnostic problem-solving and the specific structure of the solution space. If the technician strays off the tutor's path, immediate negative feedback would be provided, justified where possible with the canned rationale for evidence rules. The technician would be (a) taught in the context of problem-solving, (b) modeled as an overlay or subset of the associate's rule base, (c) instructed in the goal structure of diagnostic problem-solving, (d) have his or her working memory load minimized, and (e) have the exploration of wrong paths cut off immediately. All of the above features have been described by Anderson, Boyle, Farrell, and Reiser (1984) as the functional prescription for intelligent tutoring systems.

The Skill Integrator Interfaces

Skill integrator interfaces would have three functions: (a) to support user initiative in diagnostic problem-solving, (b) to capture the corporate memory for troubleshooting as this memory develops, and (c) to support routine maintenance event reporting. Three specific interfaces were designed to accomplish these functions for the maintainer's associate system.

The "browse" interface. The solution space in the Maintainer's Associate can be represented as a structured hierarchy of probable causes, with some indicating specific components and some indicating subsets of components. The browse interface would allow a visual representation of this hierarchy, which the user could peruse. Using a mouse or other pointing device, the user could also point to any node in the tree and call up the list of assertions which must be true in order for the system to accept that the fault could lie in the subtree beneath the indicated node. Because more than one path from the root of the diagnostic hierarchy to any given node may exist, the list of acceptable facts would be only suggestive of what actually may be the case. The user could use the browse interface to compare what he or she knows to be true to what the maintainer's associate system would accept as true for a given fault at any level of refinement.

The "jump-ahead" interface. This interface would allow the user to initiate diagnostic refinement at any given node in the solution hierarchy. While operating in the browse mode, if the user found a good match between what is known and a certain probable cause, the consultation could be started at that point. In starting at selected nodes, the Maintainer's Associate would not assert the facts it would believe. If these facts are subsequently needed, they would be automatically substantiated through the normal interaction mechanism. If the user made a poor judgment about where to begin, the system would eventually back up to the user's indicated starting point and explain that no further progress could be made with this node as the starting place.

The briefing interface. The briefing interface would have two facets: prebriefing and debriefing. Prebriefing would permit access to the maintenance information system's records on the aircraft, system, black box, or card under test. Useful information such as the component's repair history or environmental and mission correlates of the malfunction could be accessed with this interface.

The debriefing interface would be a gateway to a text file for user comments. These comments could be indexed by the node in the probable cause hierarchy at which notes were entered; and users could make comments, about any aspect of the interaction with the associate, ranging from apparent knowledge base inaccuracies to suggestions for new rules.

In a sophisticated associate, this interface would not merely accept textual input but would actively format it in accordance with the comment type. If the comment concerns the knowledge base, the system would verify this with the user and attempt to formulate the suggestion in the semantics and syntax of the rule base. Furthermore, in later developments, the briefing interface would not only accept user comments, but also request them. For example, when the user successfully solves a problem using the "jump-ahead" interface, the Maintainer's Associate would use the interface to initiate a dialogue to capture the heuristic that the technician had successfully applied and which enabled the jump-ahead.

Later versions of this interface could also serve as the technician's access point to the ground-based maintenance information system in which data about the maintenance event are collected and/or reported. The technician could input the corrective action, time taken, and other standard maintenance information upon the successful completion of fault isolation and repair.

As previously noted, the design features outlined in this chapter provide an idealized operationalization of the maintainer's associate concept. Those features that were selected for implementation and demonstration in the prototype system are described in the next chapter.

CHAPTER 3: SYSTEM IMPLEMENTATION

The basic system software and delivery hardware for the Maintainer's Associate were developed by General Dynamics, Electronics Division, as part of an independent effort. For the development of the Maintainer's Associate prototype, it was necessary for DRI to modify this basic software and design a parser. This chapter describes the additional software development and modifications, as well as the original software and hardware.

System Hardware

Software development and rule base authoring were accomplished on a Xerox 1108 personal workstation (Interlisp-D), configured with 1.5 megabytes of main memory and a 43-megabyte hard disk. The display was a large-format CRT (17" diagonal) with a high-resolution bitmap (1024 x 808 pixels). The delivery hardware provided by General Dynamics, Electronics Division, was a portable, battery-operated, briefcase-sized unit termed the "box." As shown in Figure 6, the box houses the battery pack, main processor (Intel 8086), 1 megabyte of random access memory, and a removable display/input unit. The battery pack is

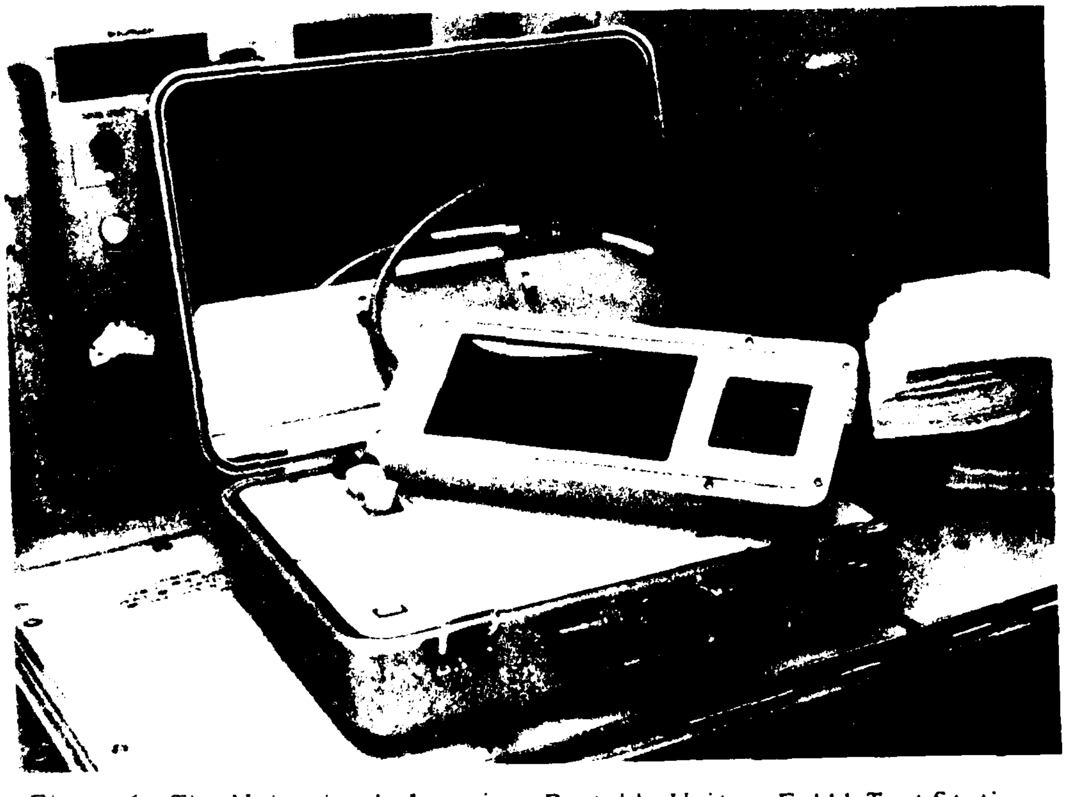


Figure 6. The Maintainer's Associate Portable Unit on F-111 Test Station Worktable.

capable of supporting 10 continuous hours of operation. The display/input unit is approximately 5" by 8"; has an electroluminescent screen with a resolution of 256 x 512 pixels and a 16-element keypad next to the screen, consisting of the digits 0 through 9; single keys corresponding to the user interfaces WHERE-TO, WHERE-FROM, HOW, MARK & RETURN; a key to move forward, labeled NEXT; and a key to move backward, labeled BACK. Software is downloaded from the development system into the box via IBM PC and RS232 connections. DRI's completed prototype Maintainer's Associate occupies a total of 118K bytes of the box's memory, including 23K bytes for the run time software, 13K bytes for the knowledge base and associated graphics, and 59K bytes for a file of expected measurements.

Supporting Software Environment

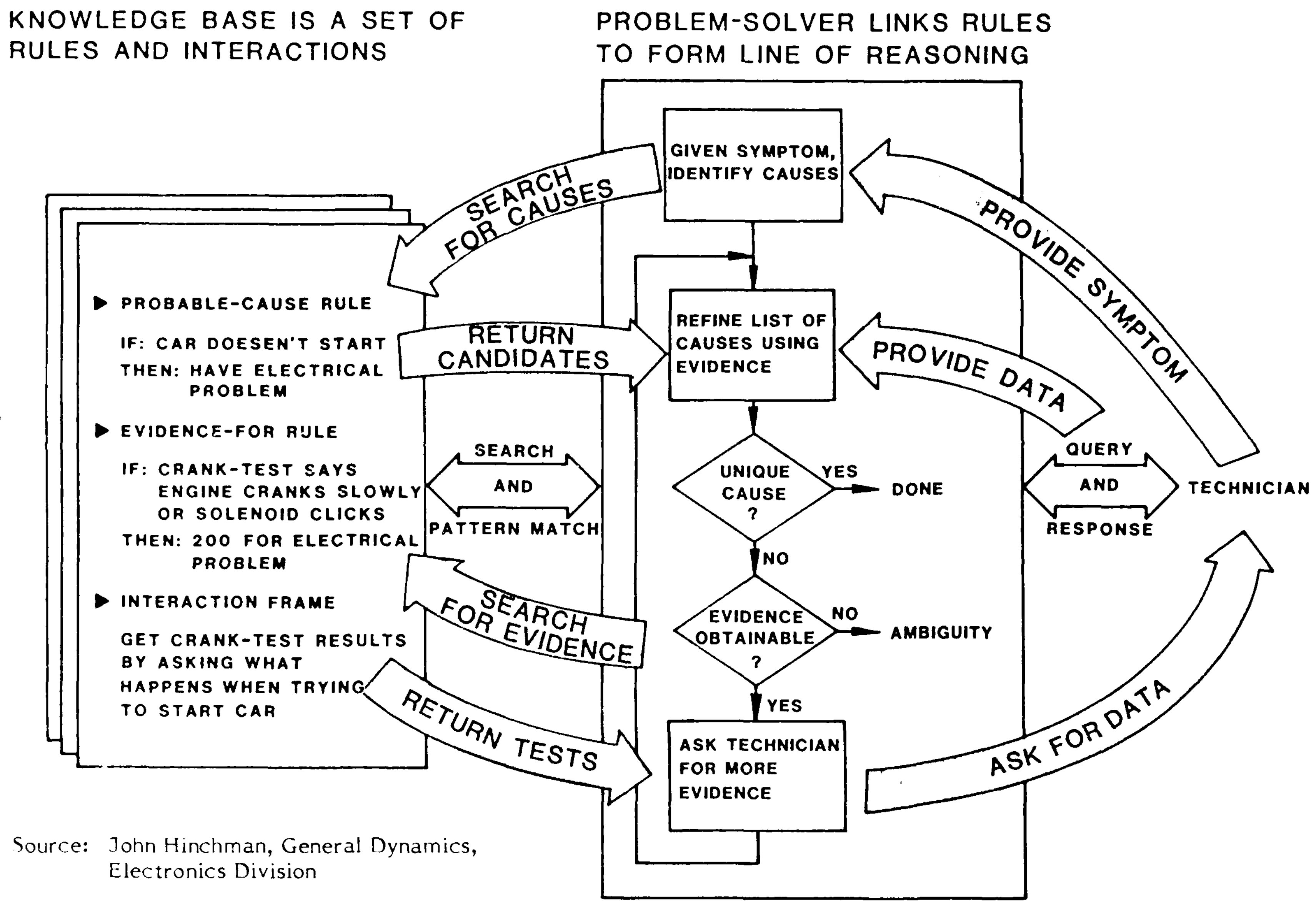
Expert System Architecture

The expert system shell used for this project is Rule-Kit (General Dynamics, Electronics Division, 1984). Rule-Kit's architecture, shown in Figure 7, uses classification problem-solving, the establish-refine approach, and a knowledge base consisting of a diagnostic hierarchy. Each node in the hierarchy contains a list of successor nodes, into which the parent is refined, and a set of rules of refinement called "evidence rules."

The basic Rule-Kit algorithm has as its objective, at each level of refinement, picking a "winner" from the successor nodes using the evidence rules contained within the parent node. This list of successor nodes is termed "the refinement list." The evidence rules ascribe weights to members of the refinement list, based on the existence of certain facts in working memory (the collection of facts developed during the course of a diagnostic session).

The first step in this process determines the existence within working memory of a fact which will cause one of the evidence rules to fire, thus assigning a specific weight to one or more members of the refinement list. After all of the evidence rules have been scanned and matched against memory, the refinement list is examined to see whether or not one of its members is now a "winner" (defined as having an accumulated weight of 100 or more points). If there is a winner, then the refinement process begins again, using the winner as the node to be refined. If there is no winner, the evidence rules are scanned again to index corresponding interaction frames which are used to request information from the user. After all the interaction frames have been collected, they are prioritized according to potential information gain. This is computed as the total points for all interaction frame outcomes ascribed by applicable evidence rules to members of the refinement list, divided by the cost of running the test and the number of outcomes.

The next step in the process is to run the first interaction frame on the priority queue. At the conclusion of the interaction, a fact is asserted in working memory corresponding to the new information developed. This fact is now


KNOWLEDGE BASE IS A SET OF RULES AND INTERACTIONS
PROBLEM-SOLVER LINKS RULES TO FORM LINE OF REASONING
▶ PROBABLE-CAUSE RULE
IF: CAR DOESEN'T START
THEN: HAVE ELECTRICAL PROBLEM
▶ EVIDENCE-FOR RULE
IF: CRANK-TEST SAYS ENGINE CRANKS SLOWLY OR SOLENOID CLICKS
THEN: 200 FOR ELECTRICAL PROBLEM
▶ INTERACTION FRAME
GET CRANK-TEST RESULTS BY ASKING WHAT HAPPENS WHEN TRYING TO START CAR
SEARCH FOR CAUSES
RETURN CANDIDATES
SEARCH AND PATTERN MATCH
SEARCH FOR EVIDENCE
RETURN TESTS
GIVEN SYMPTOM, IDENTIFY CAUSES
REFINE LIST OF CAUSES USING EVIDENCE
UNIQUE CAUSE ?
YES
DONE
NO
EVIDENCE OBTAINABLE ?
NO
AMBIGUITY
YES
ASK TECHNICIAN FOR MORE EVIDENCE
PROVIDE SYMPTOM
PROVIDE DATA
QUERY AND RESPONSE
TECHNICIAN
ASK FOR DATA

Source: John Hinchman, General Dynamics, Electronics Division

Figure 7. Schematic of Rule-Kit Architecture.

matched against the evidence rules and the appropriate rules fire, thus ascribing new points to the members of the refinement list. This process is repeated until one of the members of the refinement list is a winner or there are no more interaction frames that can be run (given a winner, the refinement process continues). If there is no winner, however, that member of the refinement list with the most accumulated points is selected.

When no further refinement is possible, it is necessary to determine whether or not the refined component is indeed responsible for the failure. If not, Rule-Kit backs up the diagnostic decision tree to a point of uncertainty and selects a different path from the one that led to the inaccurate diagnosis. By storing the refinement data in an audit stack, movement backward through the tree is controlled simply by popping data off of the audit stack. The degree of backtracking required is determined by popping the stack until a decision point is discovered which had no clear winner (i.e., no element in the refinement list with at least 100 points). The successor node that had been chosen is then eliminated from the refinement list, and the diagnostic process resumes at this level.

The Rule-Kit software employed in the Maintainer's Associate project consisted of (a) a Rule-Kit development system, (b) a Rule-Kit delivery system, (c) a Validator-Verifier, and (d) a graphics workstation. Each of these elements is briefly described in the following sections.

Development System

The development system software provides for editing and running a Rule-Kit application. It is written in Lisp and has been ported to a number of different machines, including a Symbolics 3600, a Xerox 1108, and an IBM PC-XT. Although the versions of Lisp differed for each host system, the application knowledge base was completely portable since its syntax is invariant (simply an ASCII text file). The development system consists of the Rule-Kit inference engine and a set of commands used to run consultations and to build or edit the knowledge base.

Delivery System

A streamlined version of the Rule-Kit software enabled running consultations on the portable hardware unit. This software was written in the "C" programming language and occupies 23,552 bytes. Knowledge bases developed on the development system were transferable, without change to the rule syntax, to the run time (delivery) environment for execution by the Rule-Kit run time system. The run time system compressed the knowledge base file in order to minimize memory space usage in the portable hardware unit.

Validator-Verifier

The validator-verifier is an automated version of the Rule-Kit inference engine. Its purpose is to take an existing application (knowledge base) and exhaustively construct all paths from each initial symptom, within a given range of focus, to its terminating probable cause. For each path, an audit trail is maintained that contains pertinent information used in constructing the line of reasoning from initial symptom to terminating probable cause: the interaction frames examined, the answers selected, the level of consultation, the assertions made, and evidence points given to probable causes. In addition, the validator-verifier labels all assertions according to supporting evidence linkage, whether currently linked (evidence exists at current level), later linked (evidence exists at lower level), or not linked (no evidence at any level). All audit trails are saved for interpretation by the user through the use of a number of analysis functions.

Graphics Workstation

This facility supports the construction of graphic images displayed in conjunction with interaction frames. Using a graphics table and user-friendly menu of options, graphics with associated text are rapidly developed, scaled, edited, and saved for use. Graphics are postprocessed by a data compression routine to minimize memory usage.

System Development

The Denver Research Institute developed three related software elements for the prototype Maintainer's Associate: (a) the CENPAC parser; (b) modifications to the existing Rule-Kit, specifically the glass box editor; and (c) user interface features. In order to explain the use of the prototype and the software, a scenario which illustrates typical troubleshooting procedures is presented.

A Troubleshooting Scenario

The setting for this scenario is an F-111 intermediate-level maintenance shop. A faulty UUT (in this instance a Feel & Trim Computer) is delivered by flight line personnel for diagnosis and repair. The technician begins troubleshooting by connecting the UUT via cables to the 6883 test station and initiates the appropriate automatic testing sequence. The test station, under the control of a CENPAC computer, performs a series of tests on the UUT, each designed to test a specific component of the UUT. Assume for this scenario that the testing sequence halts at test 301982. This test failure seems to indicate that the malfunction has been located. At this point, the technician disconnects the faulty UUT and re-runs test 301982, this time using the shop standard UUT known to be in perfect working condition. The test fails again, thus isolating the fault to the test station itself rather than the UUT.

In a typical maintenance shop, the technician would now use the technical orders and common manual test equipment to pinpoint the fault. With the assistance of the Maintainer's Associate, however, the technician is aided in this further troubleshooting process. The Maintainer's Associate asks the technician to make a series of tests and report the findings, and uses the answers to help isolate the malfunction. To isolate the fault, the Maintainer's Associate uses data generated by the CENPAC parser.

The CENPAC Parser

The CENPAC parser was developed in order to provide important state-specific data to the run time Rule-Kit software. At the beginning of each consultation session, the Maintainer's Associate asks the technician to enter the test number at which the test station failed. Based on the test number, the parser places in working memory the set of instantiations (expected measurements) for each generic region of the test station. For the 301982 scenario, a number of lists are placed in working memory, each of which includes: the generic region which serves as the key for the match (e.g., STIMSOURCE-OUTPUT), the signal value expected to leave the region (e.g., .08 Hz 4.0 VOLTS MOD-SIN-WAVE), and the location for measuring the signal (e.g., A4A4J4 PINS A B). A4A4 is the reference designation used by the 6883 test station documents to denote the signal generator.

These lists of information are used in the following way. In the Rule-Kit interaction frames, all references to the test station are made in terms of generic regions. The interaction frames for these regions contain variables in the message template which are bound by matches to the working memory just before the interaction frame is run. For example, an interaction frame might ask: "Check the output of the stimulus source at SIGNAL-LOCATION for this signal: SIGNAL-VALUE. Is the signal correct?" When this interaction frame is invoked, working memory is scanned for a match on the region associated with this interaction frame, STIMSOURCE-OUTPUT. When the match is found, SIGNAL-LOCATION and SIGNAL-VALUE are replaced in the interaction frame message with the specific signal location (i.e., A4A4J4 PINS A B) and the specific expected signal value (i.e., .08 Hz 4.0 VOLTS MOD-SIN-WAVE), so that the message now reads:

> "Check the output of the stimulus source at A4A4J4PINS A B for this signal: .08 Hz 4.0 VOLTS MOD-SIN-WAVE. Is the signal correct?"

The generic region instantiations that are generated by the parser could not be derived directly by decoding the test program set for a given test number as technicians do. Instead, the information had to be derived from the accumulated state of the test station at the start of each test. To illustrate how the test program set for successive test numbers yields all the information needed, consider the example scenario once again. For test number 301980, which precedes 301982, relay 10/1 is set to route the stimulus signal to the UUT; and the generic region instantiations for that test include pins and test points associated with relay 10/1. In test 301982, relay 05/2 is set to route the stimulus signal to the UUT. As tests are run sequentially until all of them complete without error or until the testing sequence halts at a failed test, the generic region instantiations for 301982 include pins and test points for both relay 05/2 and relay

10/1. Relay 10/1 was not reset after 301980, and is still available to route signals during 301982. The parser handles this problem of state accumulation by breaking the code translation process into two steps: (a) decode the test program set in order to identify the major devices used in the test and the value of the signal routed to and coming out of the UUT; and (b) use the components identified in the first step to pinpoint the signal path used in the test, thus enabling test locations to be identified along the signal path.

Figure 8 provides a detailed description of the parser process. The test program set is decoded using encoded definitions from the technical orders as seen on the left-hand side of the figure. The decoded test program set is used to simulate the test station configuration. Finally, encoded technical orders tables and a list of abstract regions are matched against the test station configuration to yield test locations and expected signal values. This process is described more completely in the following sections.

Decode the test program set. The first step in parsing, the decoding process, is relatively straightforward. For the 6883 test station, the test program sets are represented as hexidecimal codes and subcodes, which are easily distinguished. Once a code is recognized, it is simply a matter of looking up the code and its subsequent subcodes in the appropriate technical orders table to obtain the translation. For example, a portion of the test program set from test 301982 looks like this: . . . 325100 131025*325100 414058* . . . For each set of six characters, a trailing "*" indicates that the set begins with a new code; thus, in the example, 13 is a new code and 1025325100 are the associated subcodes. Subcodes always follow the code to which they pertain, and the discovery of a code in the sequential code/subcode string indicates that a different table must be used. Figure 9 illustrates the translation process. Using the proper technical orders table, it can be seen that the code/subcode string shown previously, 131025325100, provides the following instruction regarding 6883 configuration: "Set stimulus relay 05/2, transfer signal directly."

This decoder is not device-specific; that is, it contains no specific 6883 knowledge. Both the test program sets and the technical orders tables associated with each code are viewed as data. This means that with the addition of the proper code translation tables, the decoder can be used for other test stations or other state-dependent equipment.

Identify the signal path. The second parser component, the state accumulator, contains some very specific, domain-dependent knowledge. The accumulator uses global variables to keep track of the system state; these variables contain information about active stimulus sources, set or reset stimulus relays, current response relays, and currently active measurement devices. For each (sequential) test number, the decoded test program set is used to change the appropriate global variables. In this way, the accumulator identifies the test station components in use during the test. The components are then mapped across those generic regions which will be referenced by interaction frames in the prototype Maintainer's Associate. The final step is to identify the pins and test points associated with each region. A technician would do this by looking up the various components in the appropriate technical orders tables and/or schematics;

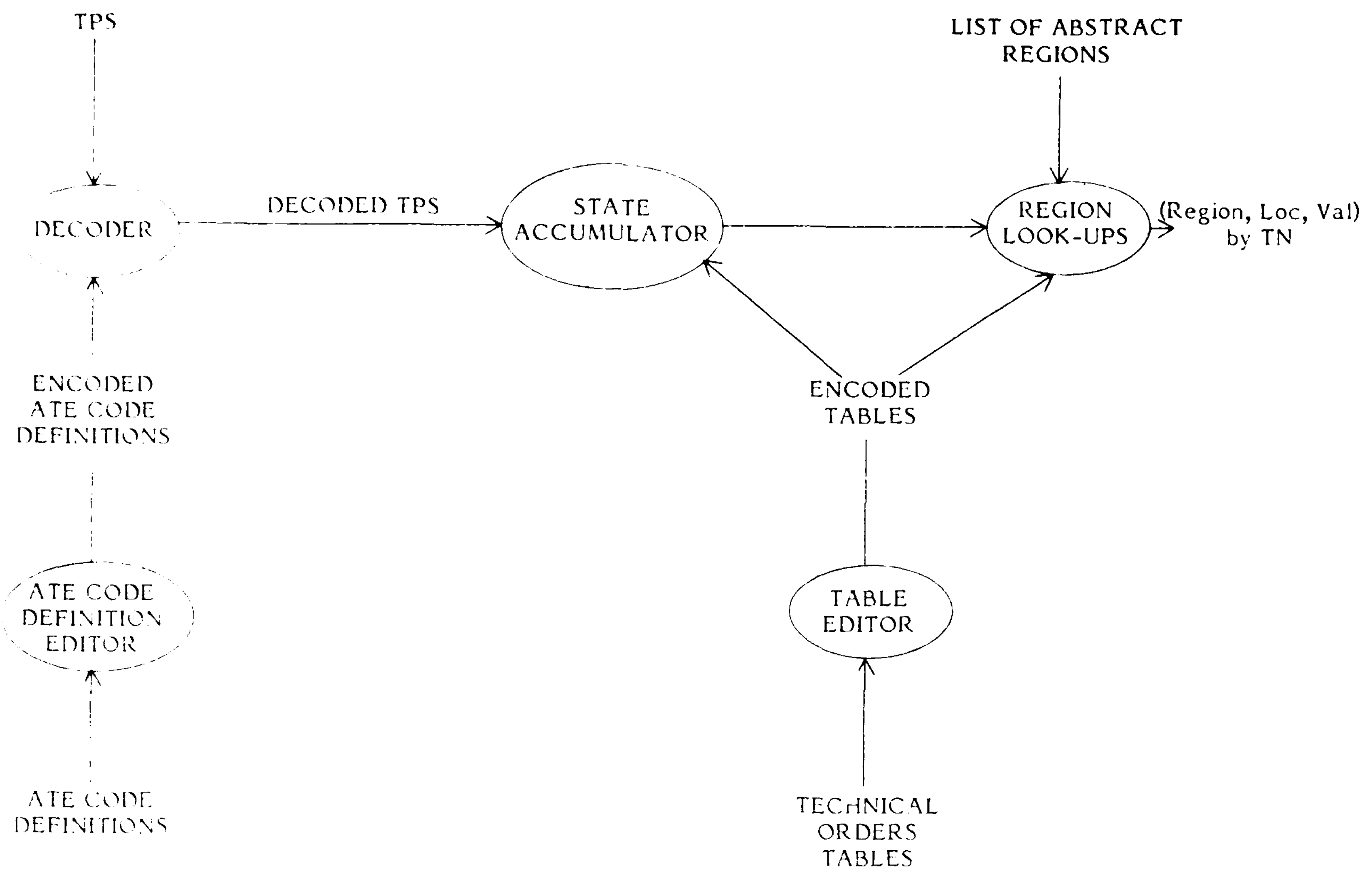
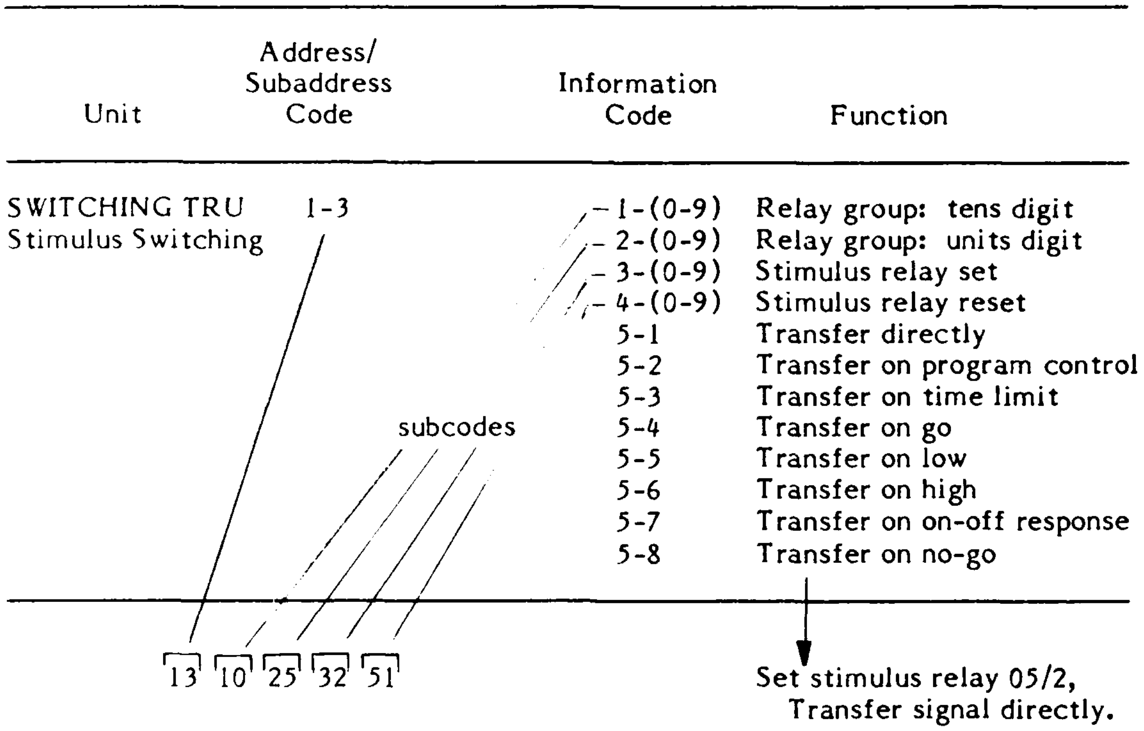


Figure 8. Schematic Representation of the Parser Process.



Unit	Address/ Subaddress Code	Information Code	Function
SWITCHING TRU	1-3	1-(0-9)	Relay group: tens digit
Stimulus Switching		2-(0-9)	Relay group: units digit
		3-(0-9)	Stimulus relay set
		4-(0-9)	Stimulus relay reset
		5-1	Transfer directly
		5-2	Transfer on program control
		5-3	Transfer on time limit
	subcodes	5-4	Transfer on go
		5-5	Transfer on low
		5-6	Transfer on high
		5-7	Transfer on on-off response
		5-8	Transfer on no-go

Figure 9. Illustration of the Translation Process for Automatic Test Programming.

the accumulator works the same way. Using indices derived from the identified components, the appropriate tables yield signal location information for each of the generic regions. Nearly 30 technical orders tables and schematics have been encoded for use during this process (see Appendix A).

It was orginally thought that the accumulator could obtain all necessary information (input and output signal values, signal source, stimulus and response relays, and the measurement device for the test) from the decoded test program set. However, although it is true that most of the tests use all of these components, there are some tests which use only the signal source and stimulus relay(s). Still other tests do use all the components, but they are identified in the test program set for a test which has already been run. Further, the expected signal value for the response side of the test loop is not always present in the test program set. To handle these variations in programming, specific rules were developed.

Multiple stimulus sources. There are two types of signal sources for the 6883: hardwired signals whose voltage never varies; and signals produced by a test station device, such as the Ratio Transformer or Signal Generator, whose voltage is determined via the decoded test program set. Hardwired signals are simply power sources which are connected to a stimulus relay. The relay acts as a gate to halt the flow of current or allow it to pass through to the UUT. When the relay is set, the hardwired signal source is active; when reset, the source is not used. In this way a hardwired signal source can be set in a given test, but used in many subsequent tests. The state accumulator uses each active hardwired signal for each test until a command is encountered (from the decoded test program set) to reset the associated relay, thus rendering the source inactive.

In the test station there are several devices that are activated by the appropriate setup commands and that may be used to generate a signal. In addition, each source is connected (just like the hardwired signal) to a specific relay. The relay must be in the proper set or reset position to allow the current to pass freely. A given test uses only one signal input from a given signal-producing device; however, a single test program number may generate numerous source/relay combinations. For example, test number 301982 instructs the Signal Generator to generate three separate stimulus signals, while the test itself makes use of only the first signal. The accumulator keeps track of each signal separately, using the first signal to set the test station state for the current test, the second for the next test, and the third for the next test after that. In addition, the signals so generated remain active as long as the associated relays remain in the proper set/reset position. This means that the three signals generated in test 301982 may be reused several times, as with the hardwired signals. The difference is that only one of the three signals may be used for each test, and the accumulator must use them in the proper sequence: first, second, third, first, and so on.

In most cases, both signal types are present in the same test. Any experienced technician would troubleshoot the paths designated by each of the source/relay combinations. The Maintainer's Associate must provide signal location and value information for all paths in order to properly duplicate the human troubleshooting process. This is accomplished by providing several instantiations for the generic regions, all of which are associated with the same test number. Test 301982 contains four such signal sources: the FCS Power Supply, the Signal Generator, a hardwired signal routed through relay 10/1, and another hardwired signal routed through relay 05/0. The accumulator recognizes those cases when more than one signal source is present and generates a set of instantiations to represent each source. Rule-Kit, in turn, provides the mechanism to query first one set, then the next, until the malfunction is found.

Variations in response signal designation. The majority of tests set a response relay which routes the output of the UUT to some measurement device. For these tests, the response relay and measurement device(s) are identified by the decoded test program set. Identification is straightforward and the accumulator needs no specialized rules to set the state of the test station; the signal value is calculated using upper and lower limit values found in the test program set. There are a number of ways, however, that this scenario may vary. In some cases, the response signal is not routed through the test station. Troubleshooting such a test requires signal location and value information for the stimulus side only (from signal source to UUT), and the accumulator need not attempt to identify the response side (from UUT to measurement device). In another case, no response relay or measurement device is designated in the test program set because the devices from the previous test are reused . Still other cases require complex arithmetic maneuvers on the part of the accumulator (or the technician) in order to determine the signal value coming out of the UUT. These complex cases were eliminated from the present prototype development effort.

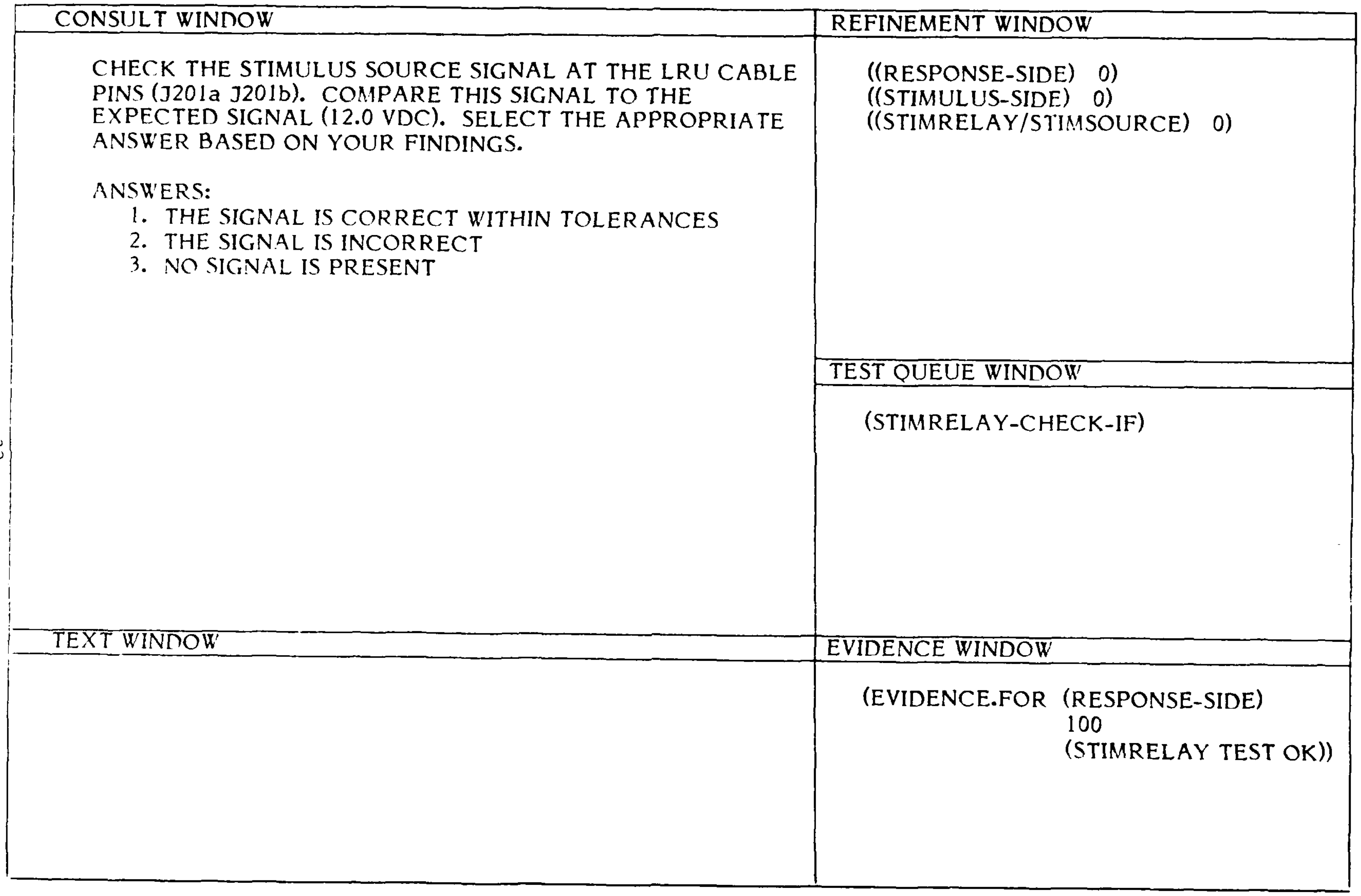
Data vs. knowledge: The final parser output. The generic regions used to capture the state of the test station for any one test are not always used for every test. In other words, although 40 potential regions have been identified, for any given test only a subset of these may be used. The troubleshooting strategy inherent in the Maintainer's Associate, however, successively refines a problem space comprised of all 40 regions. This makes it necessary for the parser to provide instantiation data for all regions, not just those which are used in the test. For regions which are used in the test, the signal location and value information are output in the form described earlier (REGION-NAME SIGNAL-LOCATION SIGNAL-VALUE). For unused regions, the system is specifically alerted to the assumption that a region that is not in use cannot be at fault. Therefore, the parser output is not data but knowledge. The parser provides the actual fact that would have been asserted by Rule-Kit had an interaction frame been run absolving the region. This fact is of the form (REGION-NAME TEST OK). When the unused region appears in the refinement list during a consultation, this fact already exists in working memory via generic region instantiation at the start of the consultation. The fact causes an evidence rule to fire, absolving the region of blame without having asked the user to make a test.

The Glass Box Editor

The glass box editor operates concurrently with Rule-Kit consultation sessions. In developing the editor, DRI provided the ability to add, change, or delete data in the knowledge base by suspending consultation and entering an editing environment. When editing is complete, consultation is resumed with knowledge of new data and the changes.

When the editor is invoked, the screen is comprised of three windows: the consult window, the refinement window, and the test queue window. Figure 10 shows all three editing windows, as well as two other windows the user may access for further information: a text window and an evidence window. The consult window shows the interaction frame message which is seen by the user during run time. The refinement list and test queue windows, which are available with the consult window, are also displayed. When activated, these two windows provide the ability to edit the associated probable cause rules and interaction frame templates. The refinement window also provides the mechanism by which the evidence rules associated with a selected probable cause can be displayed or edited. Finally, the text window is activated whenever necessary to display or edit text, such as the text message in an interaction frame.

At some point in a consultation, the user may suspend the reasoning process by entering the word "suspend" in response to the prompt from an interaction frame. Once suspended, the mouse processor keeps track of any movement or action. If the mouse moves or a button is clicked, the mouse



CONSULT WINDOW

CHECK THE STIMULUS SOURCE SIGNAL AT THE LRU CABLE PINS (J201a J201b). COMPARE THIS SIGNAL TO THE EXPECTED SIGNAL (12.0 VDC). SELECT THE APPROPRIATE ANSWER BASED ON YOUR FINDINGS.

ANSWERS:
1. THE SIGNAL IS CORRECT WITHIN TOLERANCES
2. THE SIGNAL IS INCORRECT
3. NO SIGNAL IS PRESENT

REFINEMENT WINDOW

```
((RESPONSE-SIDE)  0)
((STIMULUS-SIDE)  0)
((STIMRELAY/STIMSOURCE)  0)
```

TEST QUEUE WINDOW

```
(STIMRELAY-CHECK-IF)
```

TEXT WINDOW

EVIDENCE WINDOW

```
(EVIDENCE.FOR  (RESPONSE-SIDE)
               100
               (STIMRELAY TEST OK))
```

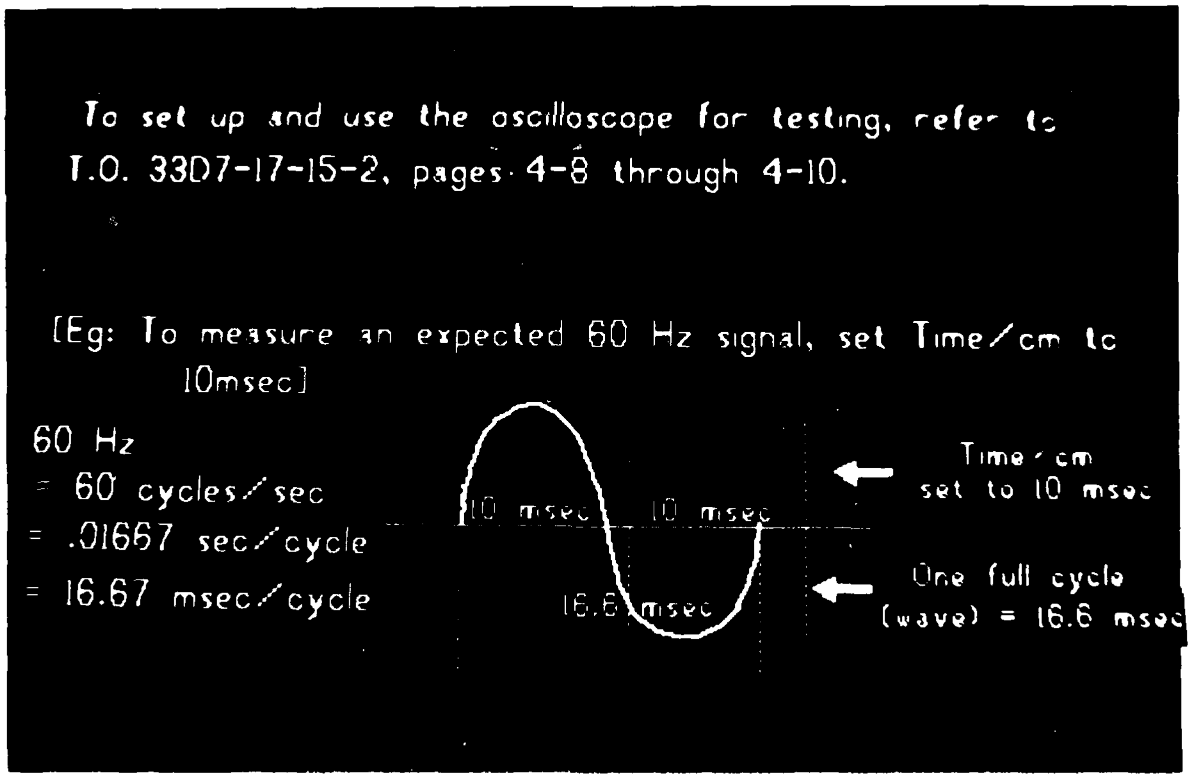
Figure 10. Display Layout of the Glass Box Editor.

processor initiates the appropriate action to be taken by the editor. Editing continues until consultation is resumed via a mouse command. This causes the refinement list and test queue to be regenerated and the interaction frame at the top of the queue to be run.

User Interfaces

The normal Rule-Kit consultation mode, in which the user responds to questions presented in interaction frames, was augmented with four user interfaces as part of the design process (see Chapter 2). These interfaces were partially implemented in the prototype Maintainer's Associate. The four interfaces are titled HOW, WHERE-FROM, WHERE-TO, and MARK & RETURN.

HOW. The HOW interface accesses engineering data and other support information, such as removal and replacement instructions and alignment or calibration procedures. Selection of the HOW key on the keypad provides the technician with a brief explanation of the required procedure. For example, Figure 11 shows the HOW display for setting up the digital voltmeter. The principal focus of the present effort was the development and use of diagnostic knowledge (troubleshooting procedures) employing AI methods, not on the development and display of procedural information. Separate projects in the IMIS program (the Computer-Based Maintenance Aids System and the Portable


To set up and use the oscilloscope for testing, refer to T.O. 33D7-17-15-2, pages 4-8 through 4-10.

[Eg: To measure an expected 60 Hz signal, set Time/cm to 10msec]

60 Hz
= 60 cycles/sec
= .01667 sec/cycle
= 16.67 msec/cycle

Figure 11. Sample HOW Frame from Maintainer's Associate Display.

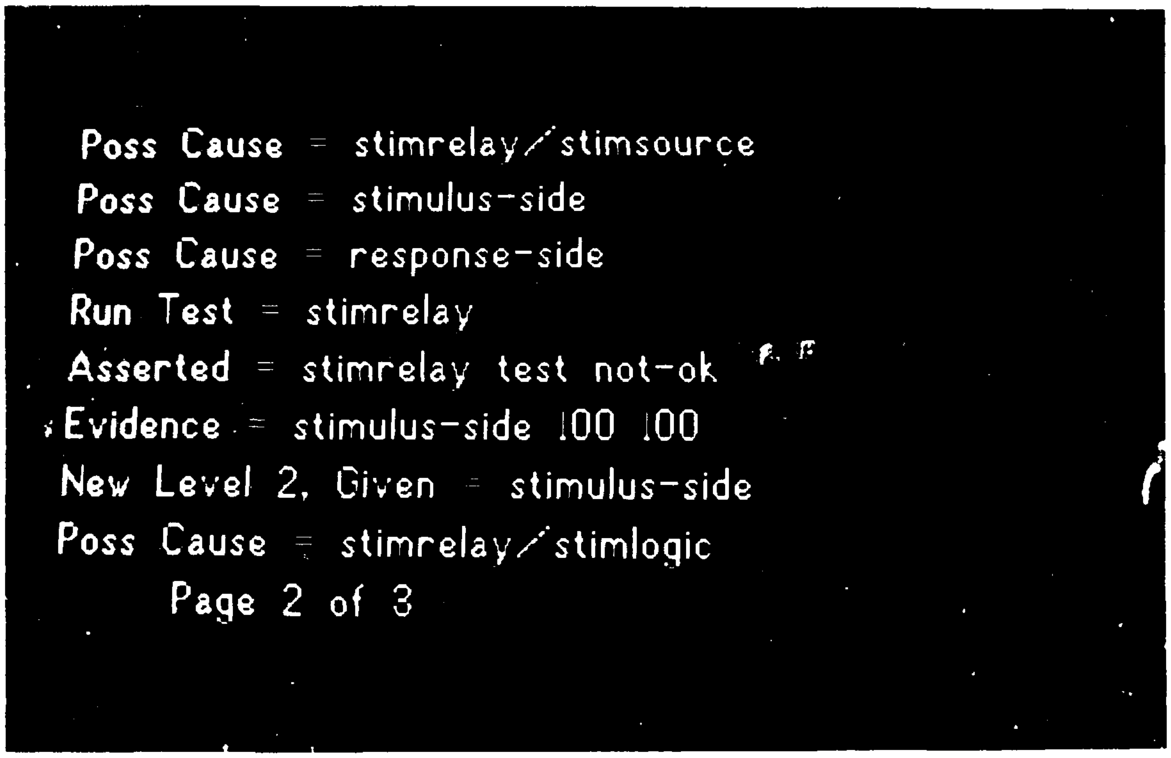
Computer-Based Maintenance Aids System) have investigated presenting and displaying this type of information. Therefore, although access to this type of information was incorporated in the prototype Maintainer's Associate, only two illustrative HOW frames were created: one for setting up the oscilloscope and one for setting up the digital voltmeter.

WHERE-FROM. In the prototype, this interface was implemented as an audit trail of the consultation in process. By pressing the key labeled WHERE-FROM on the keypad, the technician is shown the path of the consultation up to that point. When Rule-Kit establishes and refines a level because the correct assertions already exist in working memory, the technician does not see an interaction frame. The audit trail records the interpretive process regardless of the presentation of interaction frames. Thus, the parser's assertions may cause the interpreter to find a winner without having to ask anything of the technician and the audit trail may include entire establish-refine cycles not apparent in the consultation's series of interaction frames.

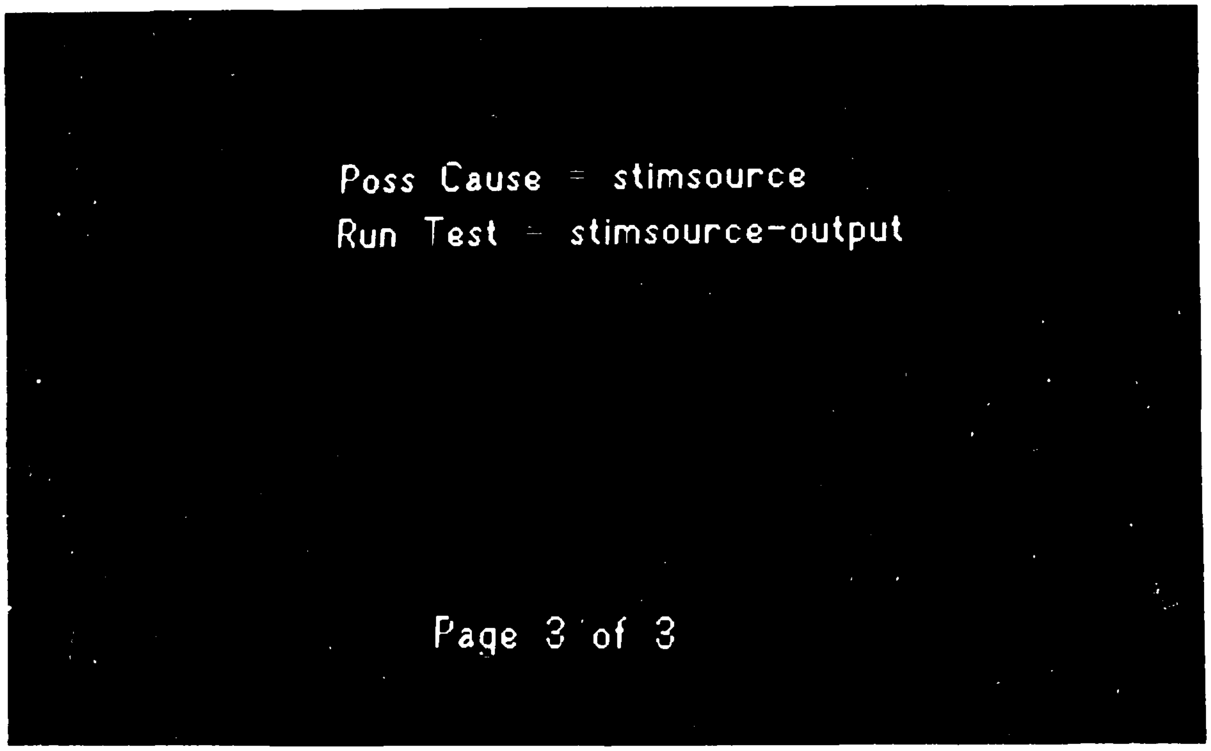
Figure 12 shows two sample screen displays created by the WHERE-FROM option. The screen is organized by levels of refinement. The first level is a list of all assertions made by the parser at the beginning of the consultation; that is, those facts that were asserted, based solely on the test number at which the fault was manifest. Each subsequent level of refinement presents information corresponding to the interpretive process of the Rule-Kit inference engine. For example, in Figure 12 these include the refinement list (a list of probable causes); the interaction frame to be run, together with a list of the points ascribed to probable causes for each outcome; the outcome of the interaction frame; and the name of the winning frame.

WHERE-TO. Selection of the WHERE-TO feature results in a display which contains information on the expert system's pending processes, as shown in Figure 13. This interface is intended for situations in which an interaction frame asks the technician to make a measurement and report the results. If the technician has questions (e.g., "How are you going to use that information?" or "Why are you asking me that question now?"), this display explains the name of the current interaction and its cost; the name of the currently instantiated parent; and a list of the possible outcomes of this test, along with the number of points to be ascribed to each probable cause. With this information, the system states: "I have determined the fault to lie within this probable cause and the following test is requested to help me assign points to each of these probable causes now under suspicion."

MARK & RETURN. This interface is a first-approximation of a capability of the Maintainer's Associate design which enables the user to "browse-and-jump-ahead." When activated by pressing the MARK & RETURN key on the keypad, a small flashing box appears in the corner of the display to indicate that the interface is engaged. MARK & RETURN functions like a bookmark. It allows the technician to move ahead in the consultation and answer questions without actually making the measurements. During this browse of the consultation process, WHERE-FROM and WHERE-TO are also active. Thus, in each successive interaction frame, the small flashing box lets the technician know that



```
Poss Cause = stimrelay/stimsource
Poss Cause = stimulus-side
Poss Cause = response-side
Run Test = stimrelay
Asserted = stimrelay test not-ok
Evidence = stimulus-side 100 100
New Level 2, Given = stimulus-side
Poss Cause = stimrelay/stimlogic
      Page 2 of 3
```

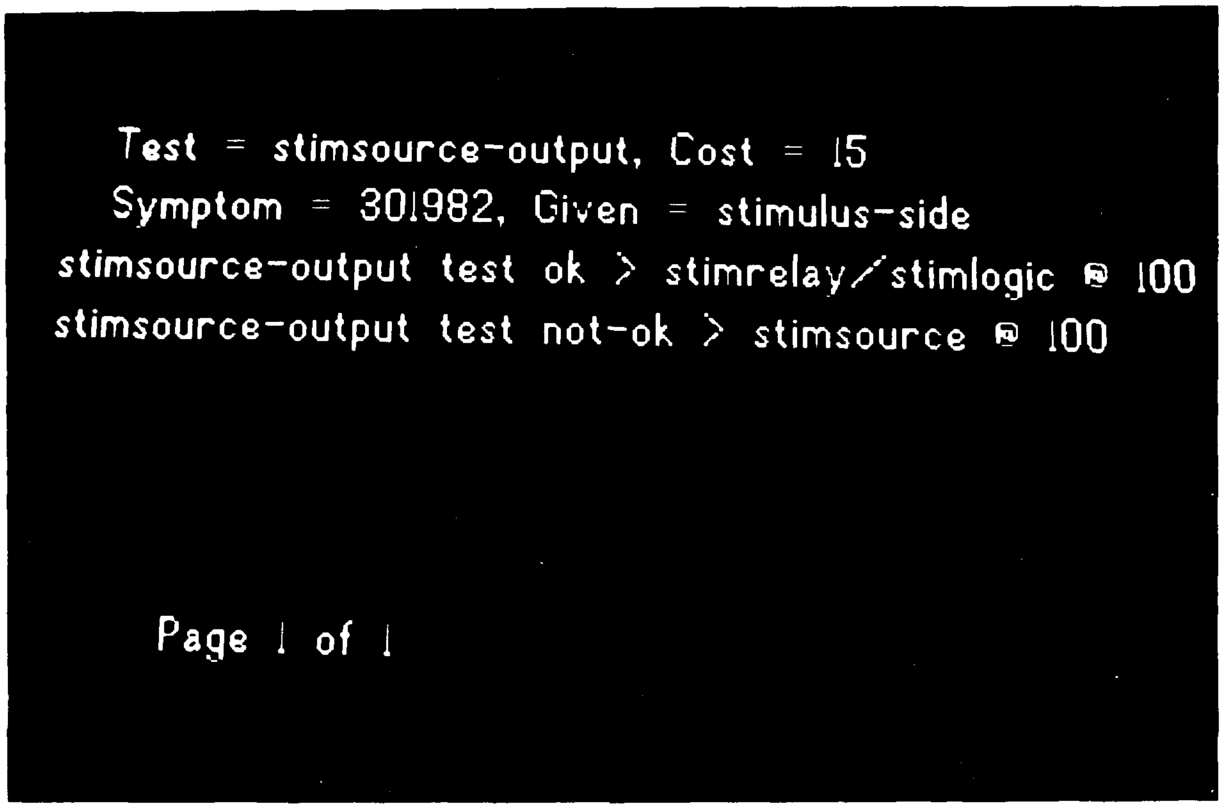


```
Poss Cause = stimsource
Run Test = stimsource-output

Page 3 of 3
```

[illegible] 2. Sample WHERE-FROM Frames from Maintainer's Associate Display.

when the MARK & RETURN button is pressed again, the system will reset the consultation to the point where MARK & RETURN was originally requested by the technician.


```
Test = stimsource-output, Cost = 15
Symptom = 301982, Given = stimulus-side
stimsource-output test ok > stimrelay/stimlogic @ 100
stimsource-output test not-ok > stimsource @ 100

Page 1 of 1
```

Figure 13. Sample WHERE-TO Frame from Maintainer's Associate Display.

CHAPTER 4: KNOWLEDGE ENGINEERING

Introduction

Knowledge engineering has been described as more of an art than a science. This is due to the complexity inherent in extracting, understanding, and representing the knowledge of experts in the development of a computer-based system. It is no surprise that the task of knowledge acquisition (i.e., extracting knowledge from an expert) is a major bottleneck in the development of expert systems. Attempts have been made to structure the knowledge engineering task, to automate the process with expert system development tools and CAD/CAM processes, even to eliminate it by developing sophisticated interfaces that would allow subject-matter experts (SMEs) to develop rule bases for expert systems directly.

In this chapter, the knowledge engineering process used by DRI in the development of the Maintainer's Associate is described. Following a brief description of task purpose and overall approach, this chapter presents the specific processes involved in the two-level development of the Maintainer's Associate rule base and discusses the results of the effort and its relation to relevant knowledge engineering issues.

Background

Given a choice, technicians tend to rely on symptom-based approaches to solve troubleshooting problems. The symptom-based approach, in which the technician looks for a match between the symptoms of a past fault and the current symptom, is easily incorporated into expert systems. Although more time-consuming, specification-based methods are usually more accurate. Relying on an analysis of the function and structure of the device under consideration, specification-based methods include exhaustive searching for faulty components (practical only for relatively small sets of possible faults) and reference to a normal functioning model. Using the latter method, the technician constructs a mental model of the device in a normal functioning state and develops a set of hypotheses like "what would happen if," then discrepancies between the device under test and a normal functioning device lead to identification of faulty components. Other methods, including half-split, bracket, and uncertainty reduction techniques, are enhancements to basic methods and help reduce the number of suspects in the ambiguity group.

As discussed in Chapter [illegible], electronic troubleshooters use both heuristic ([illegible]) and [illegible] (specification-based) approaches to isolate faults in equipment. Expert systems, in general, have used one or the other approaches as a base for the development of the rule set. One goal of the knowledge engineering component of this project, therefore, was to capitalize on the interrelatedness of the two approaches and use both in the development of the rule base. The method was to compile specification-based knowledge into a diagnostic test tree and overlay symptom/fault information as additional branches or tests.

Assumptions

The troubleshooting environment for this prototype is a complex environment. Electronic maintenance equipment is inherently complex, and new developments in equipment design have increased the complexity. In order to test the concept of applying expert systems to this environment, several assumptions were necessary.

Single fault assumption. In the real world of automatic test equipment (ATE) modeled in the prototype, faults in the electronic assemblies and subassemblies can trigger secondary faults which must be located and repaired before full system capability is restored. Development of a rule base to account for all possible combinations of faults was considered a task too expansive for the present effort. Therefore, an underlying assumption of this task was that only a single fault is present in the equipment. Identification of that single fault was the objective of the diagnostic tree.

Nonintermittency assumption. A second assumption was that the fault manifest in the system is nonintermittent; that is, it is a hard fail which is present at the same point on each rerun of the test sequence. An intermittent fault, which may or may not be present on repetitive tests, leads to inconsistencies in the diagnostic process. Although intermittent faults are common and present a serious maintenance problem, the expert system developed for the Maintainer's Associate was not intended to address this problem.

Test number associated fault assumption. A sequence of tests is automatically run by the ATE and interrupted upon discovery of a failure in the process of automatic fault isolation. Each test in the sequence of tests is identified by a test number. It is possible that a failure of a component can occur at any time in any part of the ATE, even when that component is not involved in the particular test which is being run. An occurrence of a fault under these conditions would be extremely difficult to identify, since the expert system developed for the Maintainer's Associate is keyed on the decoding of the programming associated with the test number which fails. Therefore, these haphazard faults were not considered in the system; only components associated with the areas of the ATE which are currently being used for a specific test were considered in the suspect set.

Approach

In order to accelerate the knowledge engineering process, the proposed problem domain was divided into two separate levels, and rule base development proceeded at each level in parallel. The first level consisted of the rules which isolated the malfunction to a particular TRU in the 6883 test station; the second level focused on the identification of a specific faulty component within the Flight Control Simulator (FCS) adapter TRU. Because levels are hierarchically related (Level II is determined by Level I rules), a generic approach was adopted for Level I which allowed independent Level II development. Thus, two somewhat different knowledge engineering strategies were employed.

Both approaches relied on fairly conventional sources of knowledge, primarily technical documentation and SMEs. The two SMEs for this effort were 7-skill-level avionics instructors from Air Training Command who were responsible for 6883 and related automatic test equipment (ATE) at Lowry Air Force Base, Colorado. Two supervisors from the intermediate-level maintenance shop at Cannon Air Force Base, New Mexico, also contributed technical expertise. All of the SMEs were responsible for providing information on 6883 operation, troubleshooting strategies, training objectives, and field maintenance activities.

Level I Development

Level I consisted of the rules and interaction frames necessary to isolate malfunctions to the TRU level. This task required: (a) development of a test loop which would be applied for all troubleshooting scenarios; (b) identification of a troubleshooting strategy which reflected the interdependencies and interconnections of the components of the test station, the training and performance objectives of classroom and on-the-job training, as well as skilled technician rules-of-thumb where appropriate; and (c) development of a rule set which captured all of the necessary information into the Rule-Kit architecture.

The test loop shown in Chapter 1, Figure 3, is a simplified model of the ATE which presents signal source, signal switching, routing to the unit under test, and signal measurement. For the development of the rule base for Level I, a more complete and complex representation of the ATE was required because troubleshooting to the TRU level required representation of the possible set of TRUs in the schematic, as well as the interconnection of the TRUs via cabling and wires.

This enhanced test loop was developed from technical documentation, SMEs, and prior knowledge of the ATE. Several schematics of the ATE were available in the technical orders, but none presented the test station in a way which would easily lend itself to development of a troubleshooting strategy or to development of a set of rules. The existing schematics were, therefore, synthesized into a single schematic with sufficient complexity to adequately represent the ATE as perceived by an SME, while confining the level of complexity to that which would be familiar to the target audience. This synthesized test loop schematic was shown to both instructor and field SMEs in order to define the relations of the test loop areas which represented specific TRUs and their interconnections. The final test loop schematic is shown in Figure 14.

Establishment of the test loop schematic led to development of a troubleshooting strategy which reflected the troubleshooting approaches of all

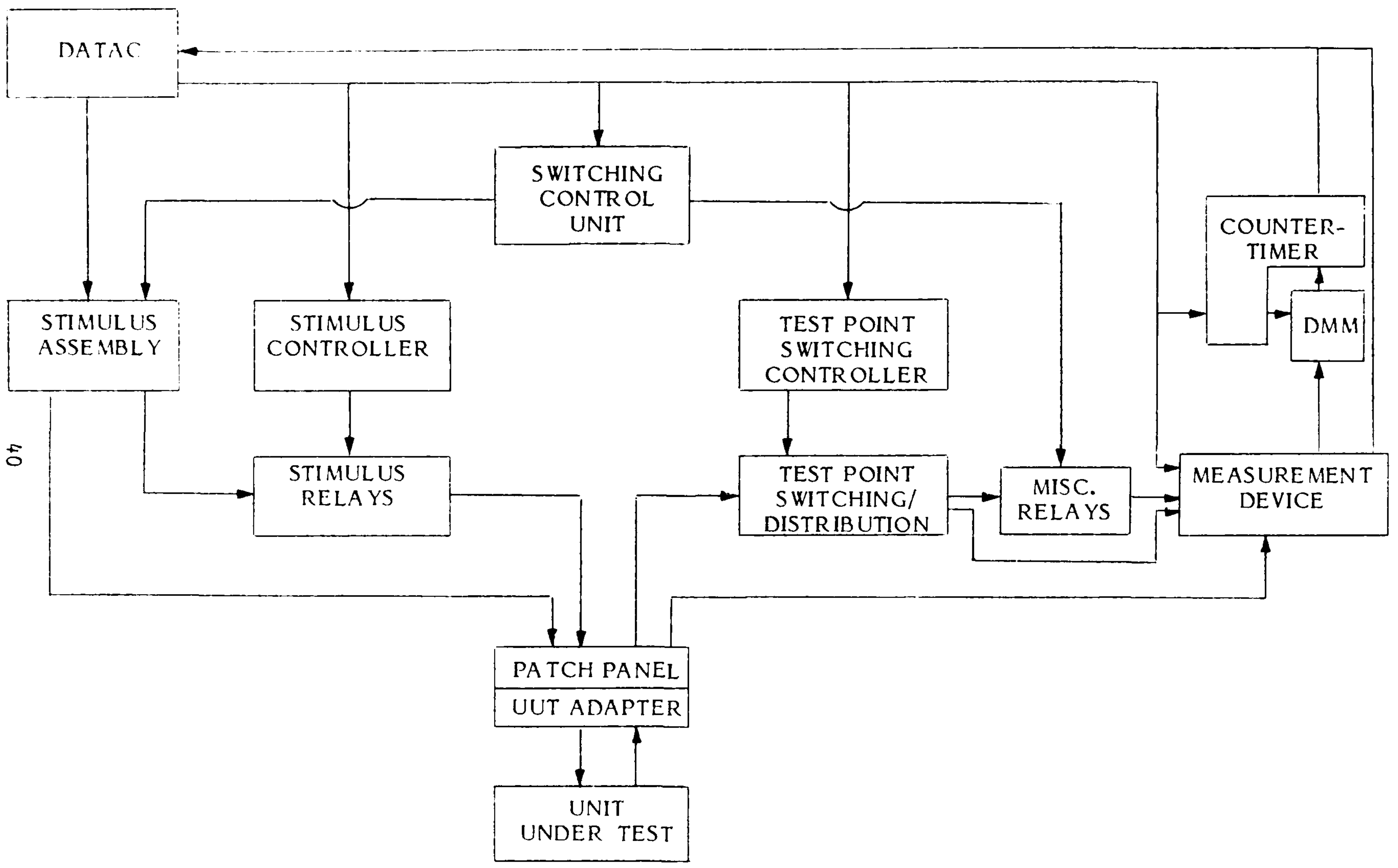


Figure 14. Test Station Dependency Model.

four SMEs. Reaching a consensus regarding troubleshooting starting points and test/measurement sequences was critical to the process. When discrepancies arose among SMEs, justification was sought for each position and a resolution reached on the best alternative. This process also helped fine-tune the test loop schematic. Both standard procedural approaches and individual SME heuristics were sought to develop the diagnostic tree.

Establishing the test loop schematic and diagnostic tree facilitated Level I rule base development. Using an establish-refine approach, the rules incorporated a starting point general to all troubleshooting problems, and a diagnostic tree of ambiguity groups resulting from test outcomes. The diagnostic tree consists of levels of refinement and nodes associated with each level. The nodes in the tree are either virtual (representing a set of actual nodes or test station components) or actual (representing a specific test station component). A part of the diagnostic tree developed for the stimulus side of the 6883 test station is shown in Figure 15. The complete diagnostic tree was based on the structure and function of the ATE, the ease of testing, the cost of testing, and the likelihood of component failure.

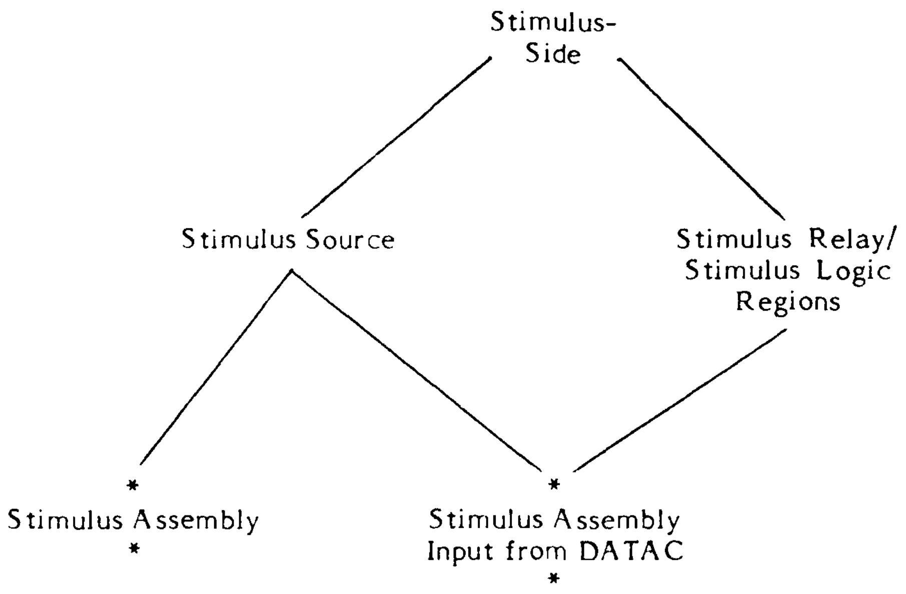

Stimulus-Side
Stimulus Source
Stimulus Relay/ Stimulus Logic Regions
*
Stimulus Assembly
*
*
Stimulus Assembly Input from DATAC
*

Figure 15. Partial Structural Representation of the F-111 6883 Converter/Flight Control Systems Test Station, Showing Relationship of Actual (*) and Virtual Nodes.

Level II Development

Level II of the knowledge base development consisted of rules that directed fault isolation within the FCS Adapter. Unlike Level I, rules at this level were specific to the FCS drawer and were not designed to apply to additional

TRUs. Development of this rule base was relatively straightforward and can be described best as a five-stage cooperative process between knowledge engineers and SMEs. First, a hierarchical structural representation of the FCS was constructed, based on the schematics, reference designations, and parts breakdowns provided in the technical orders. Figure 16 shows part of this structural breakdown for the FCS. Second, using the functional description and

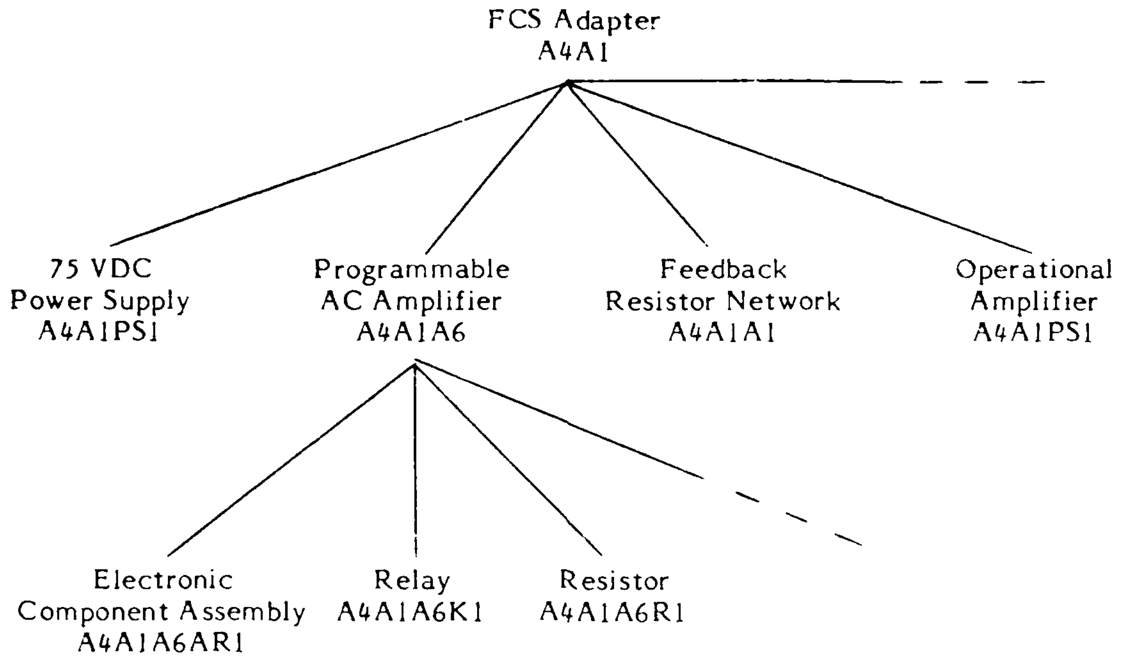


Figure 16. Partial Structural Representation of the FCS Adapter, Including Reference Designations.

block diagrams, a functional representation was developed. For the FCS, three basic functions were identified and are shown in Figure 17. The third stage required reconciling these two representations into a single system model which related structural components in a functional hierarchy. Although this approach was generally consistent with the reference designations, some components (e.g., chassis-mounted parts) were redefined to facilitate this process. Fourth, a troubleshooting tree was constructed from the system model using the following basic guidelines in addition to the basic research assumptions:

1. At the point where the FCS troubleshooting is initiated, the fault has been unambiguously isolated to the FCS.

2. Each step in the troubleshooting process distinguishes at least one alternative in the next level of the system hierarchy.

3. The order in which tests are prescribed is dictated by factors that include cost of test, likelihood of failure, and ease of repair.

4. The level of detail represented in the final tree extends only to the first replaceable/repairable component; subcomponent failures are not identifed.

Figure 18 shows a portion of the resulting test tree for the FCS which is expanded to include decision points that are internal to the Maintainer's Associate system, as well as those requiring technician input. The final stage in the knowledge engineering process consisted of translating the troubleshooting tree into individual rules for entry into the prototype knowledge base. Each node corresponds to one or more interaction frames in the system.

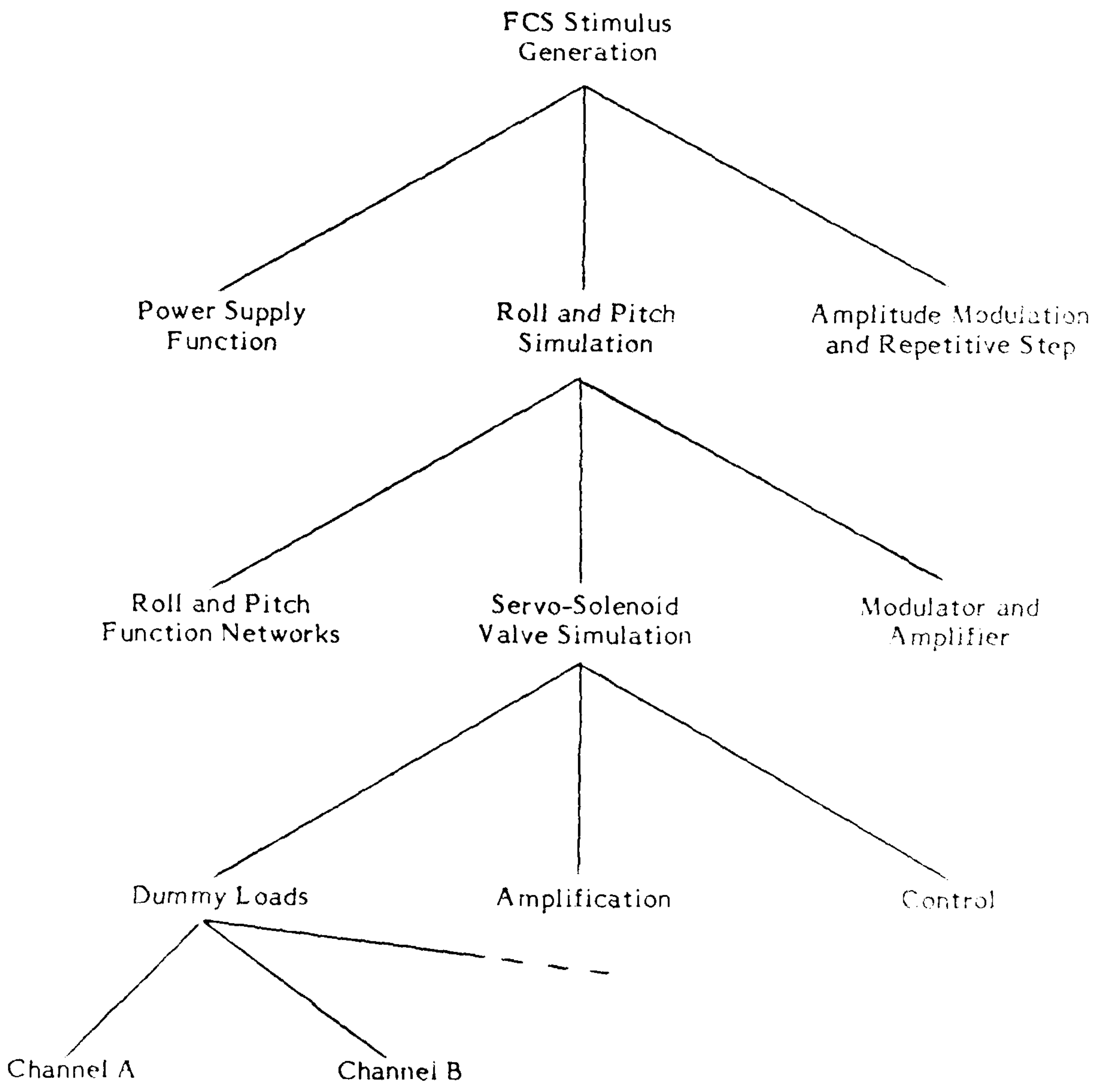

FCS Stimulus Generation
Power Supply Function
Roll and Pitch Simulation
Amplitude Modulation and Repetitive Step
Roll and Pitch Function Networks
Servo-Solenoid Valve Simulation
Modulator and Amplifier
Dummy Loads
Amplification
Control
Channel A
Channel B

Figure 17. Partial Functional Representation of the FCS Adapter.

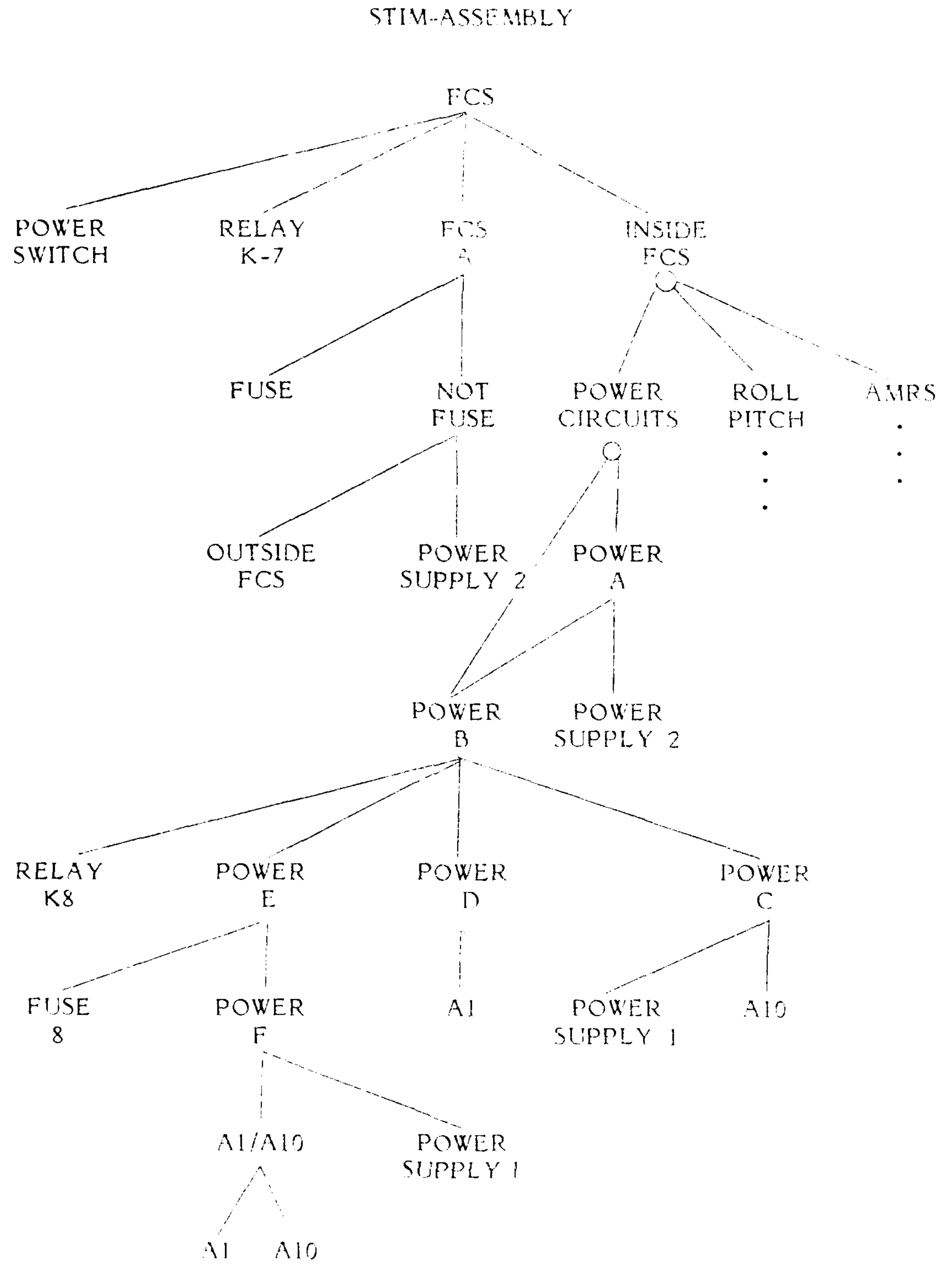


Figure 13. Partial Test Tree for Level II of the Maintainer's Associate Knowledge Base. Circles are used to identify specification-based decisions that require no user input.

Results and Discussion

For the current Maintainer's Associate prototype, DRI developed a total of 52 interaction frames, 130 probable cause rules, and 130 evidence rules. Of these rules, approximately 60% were associated with the generic Level I of the knowledge base, and the remaining 40% were associated with Level II. If it is assumed that the FCS drawer is fairly typical of all 6883 TRUs, subsequent expansion of the prototype problem domain to include additional components can be estimated at approximately 125 rules (total probable cause and evidence) for each new TRU.

More important than the number of rules, however, are the results of the knowledge engineering process in terms of the nature of the rule base and its structure. That is, in what ways did the outcomes of this process support or disclaim conventional expectations regarding several elements of the knowledge engineering task, including the role of subject-matter experts, the representation of troubleshooting test loops and strategies, and the types of rules that result? Admittedly, the Maintainer's Associate rule base is likely to represent only a subset of the actual troubleshooting processes that might be undertaken by experienced technicians, because SMEs often employ more information than they report or are even aware of using. However, the consistency of information obtained from the four SMEs contributing to the data base suggests that this effort has accurately reflected troubleshooting in avionics maintenance. For this reason, a number of issues which have general implications beyond the present project are addressed. These include approaches to device modeling, the use of heuristics, and the selection and role of SMEs.

Device Modeling

It is generally assumed that an experienced technician's knowledge in a given troubleshooting situation takes the form of a mental model of the device under test. These models are not isomorphic representations of device topography; but rather, they are composed of a number of interrelated and overlapping structures that are often hierarchical in nature. As is evident from the preceding descriptions of Level I and II development, the prototype Maintainer's Associate rule base shares many of these characteristics. For example, rules are hierarchically organized; they include both structural and functional relationships, and are not entirely parsimonious in terms of the devices represented.

However, there were several aspects of the experts' mental models that were not easily captured by the knowledge engineering process and Rule-Kit architecture. These ambiguities presented special problems to the knowlege engineering task and are briefly described. First, because a single representational format was imposed, the resulting rule base was limited to only the most critical interdependencies. Although the system allowed structural, operational, and functional information to coexist in the rule base, a comprehensive representation of all three was not practical.

Second, it was sometimes difficult to restrict the SMEs to the assumptions of the prototype and the limits of the selected problem space. For example, since relays are a common source of malfunction, SMEs sometimes skipped directly to that specific level of detail, even though an intermediate level, such as a component assembly, was more appropriate in view of the Maintainer's Associate design and the technician's task. A third problem, related to the previous two, was that arbitrary limitations on the system model did not always correspond with the meaningful levels used by SMEs. Therefore, flexible criteria were employed so that the resulting rule base was comprised of meaningful units at various levels of structural detail. Fourth, it is possible for an overlap of functional, structural, and operational information to occur. For example, fuses located on the TRUs of the ATE can be both indicators of faults (symptoms) and faults themselves. That is, a blown fuse implies a functional irregularity in the test station and until it is replaced prevents normal operation of the test station. In addition, a blown fuse may imply a fault in the TRU with which it is associated or in another TRU which is sending a (faulty) signal to the TRU housing the fuse. Finally, it was found that the boundaries between device regions or components could not be easily represented in the Rule-Kit format. Cabling and other connections are common sources of problems in the 6883 test station, yet malfunctions of this sort lie in the interface between regions, and, therefore, cannot be attributed readily to a single component. For example, lack of a firm interface between two pins can result in a test failure, even though none of the pins is broken or in need of replacement. The space between them, if it is identifiable as an element, is at fault.

One solution to this interface problem is to consider each to be a distinct component that, when operating properly, is completely passive with respect to the signal being propagated. However, carried to extremes, incorporation of rules to address these interface "regions" could cripple the system. A second solution would be to arbitrarily assign interfaces as input or output elements of specific TRUs or components, rather than as distinct components. Although certainly feasible, the successful use of this solution requires extensive reliance on SMEs to consistently assign individual interfaces. In the present effort this issue of handling faults in interface areas did not prove to be a serious concern. Most interface problems occur in the attachment of the LRU adapter and LRU to the test station, and these problems would be identified and corrected during the substitution of the shop standard LRU. Secondly, troubleshooting requires the removal of cables and other connections to perform signal measurements. During this process, faults related to poor interface connections would be purposely or incidentally identified and corrected. Finally, in the opinions of the project SMEs, the likelihood of an interface connection fault within the ATE is small. Internal component interconnects are rarely if ever manipulated, and that limits the opportunity for misalignment of pins.

Heuristics

A particular type of rule often prominent in the troubleshooting literature is the heuristic or rule-of-thumb. Heuristics are consistent with the technician's general troubleshooting approach and dictate jumping ahead to likely solution, rather than systematically considering all alternatives. Sometimes the likely solution is incorrect, of course, and the technician must backtrack and try a different alternative. It was anticipated that heuristics would play a significant role in the development of the Maintainer's Associate rule base because experts often report using them to identify malfunctions. As a result, jumping and backtracking capabilities were incorporated into the Maintainer's Associate system to accommodate rules of this type (see Chapter 3).

However, the SMEs consulted in this project made almost no use of heuristics even though they were encouraged to do so. A number of explanations for this unexpected finding were considered. One possibility was that, since the primary SMEs were instructors rather than field technicians, their approach to troubleshooting was essentially pedagogical, and thus sytematic, rather than pragmatic. Furthermore, since neither instructor SME had substantial field experience, they may simply have been unaware of useful heuristics. Constraints on field maintenance such as time, personnel, and operational readiness, may also provide a possible explanation, because these factors are not evident in the school environment. Finally, it was thought that the particular problem domain selected for the prototype, especially the FCS, might be atypical with respect to the use of heuristic rules. To examine these explanations, the Cannon Air Force Base SMEs were consulted. They could suggest no additional heuristics for the rule base and noted that heuristics were typically device-specific or aircraft-specific rather than applicable at a general systems level. That is, particular test stations or aircraft have idiosyncratic malfunctions that recur under certain conditions; and in these specific situations, heuristics are particularly effective.

Another type of heuristic that is often cited in the troubleshooting literature involves the use of patterns of information (or symptoms) by the technician. In this knowledge engineering effort, these types of rules were also not in evidence. SMEs generally relied on a stepwise establish-refine approach by focusing on a single piece of data at any one time, and patterns of information only developed sequentially in the troubleshooting process. The reasons for this are unclear, but it is likely that the perceived format of the Maintainer's Associate rule base, the digital nature of the automatic test equipment, and the assumptions regarding the types of allowable malfunctions all played a role.

Subject-Matter Expert Selection and Use

As previously noted, both training and maintenance shop environments provided SMEs for this work. The training instructors had extensive experience with the 6883 test station, but limited field repair experience. Thus, the experience of the instructor SMEs was aligned with an operational rather than a repair/troubleshooting context. On the other hand, the SMEs from Cannon AFB had substantial field experience and brought the opposite perspective--less emphasis on representational elements which may be important to the novice technician and greater emphasis on the components particularly useful to troubleshooting. This difference had subtle impacts on the knowledge engineering process in that the development of the test loop schematic required substantial review, even extensive modification, to reconcile operational representativeness with maintenance/repair modeling.

The potential for discrepancies between the perspectives or mental models of instructors and field technicians raised the knowledge engineering issues of SME reliability and validity. Although the knowledge engineer can seek a consensus among SMEs on specific points, lack of personal experience with either instruction or maintenance can limit the knowledge engineer's ability to resolve these larger questions. Once the model of the test loop and the associated troubleshooting strategy had been modified, all four SMEs agreed that it was an accurate representation. This implied that Air Training Command's troubleshooting training reflects field performance needs. However, some disagreement occurred in the area of domain knowledge. Instructor SMEs were more familiar with all the areas of the test station; the field technicians knew more about specific areas and component failure rates, which were important to developing the troubleshooting approach.

For a particular development effort, the decision to use multiple experts versus a single expert is most likely to be a practical one. In this project, multiple experts were used because of their availability and because the combination of instructor and field maintenance expertise was considered optimal in development of the troubleshooting strategy and rule base. In practice, the selection of one or more experts will depend on the availability of a single recognized domain expert whose information is considered reliable and comprehensive, the preference of the knowledge engineer for varied perspectives, or the ability of the knowledge engineer to establish multiple working relationships. The experience of this effort indicated that even with multiple experts, each becomes, in a sense, a single expert because each exhibits expertise in a particular area or subset of the domain. Conflicts that arose among the SMEs were resolved through discussion and consensus.

Conclusion

Although the approach to the Maintainer's Associate prototype knowledge engineering task was separated into two levels to address two different aspects of the rule base (the generic test loop rule set and the TRU-specific rule set), many of the same elements were involved in the development of the rule base for each level. A schematic, representative of the equipment and derived from a synthesis of existing schematics and through discussion and review with SMEs, was developed on which to build a troubleshooting strategy. The troubleshooting strategy was refined, again through discussion with SMEs, and the rules which reflected the strategy were developed. These rules were reviewed several times, especially in establishing test costs and sequences, and resulted in the final combined rule set.

In the process, it was discovered that limitations of the expert system architecture used in the present effort resulted in constraints on the ability to translate the SME system model to the rule base. In addition, heuristics, frequently given emphasis in expert system literature, may not be suitable to all systems. Their main importance may be in systems modeling specific aircraft or test stations, where modeling of idiosyncracies of the equipment would facilitate troubleshooting.

Finally, the use of multiple experts did not present significant problems regarding system modeling or troubleshooting strategy. The opportunity to incorporate both instructor and field technician perspectives was very valuable.

CHAPTER 5: SYSTEM DEMONSTRATION

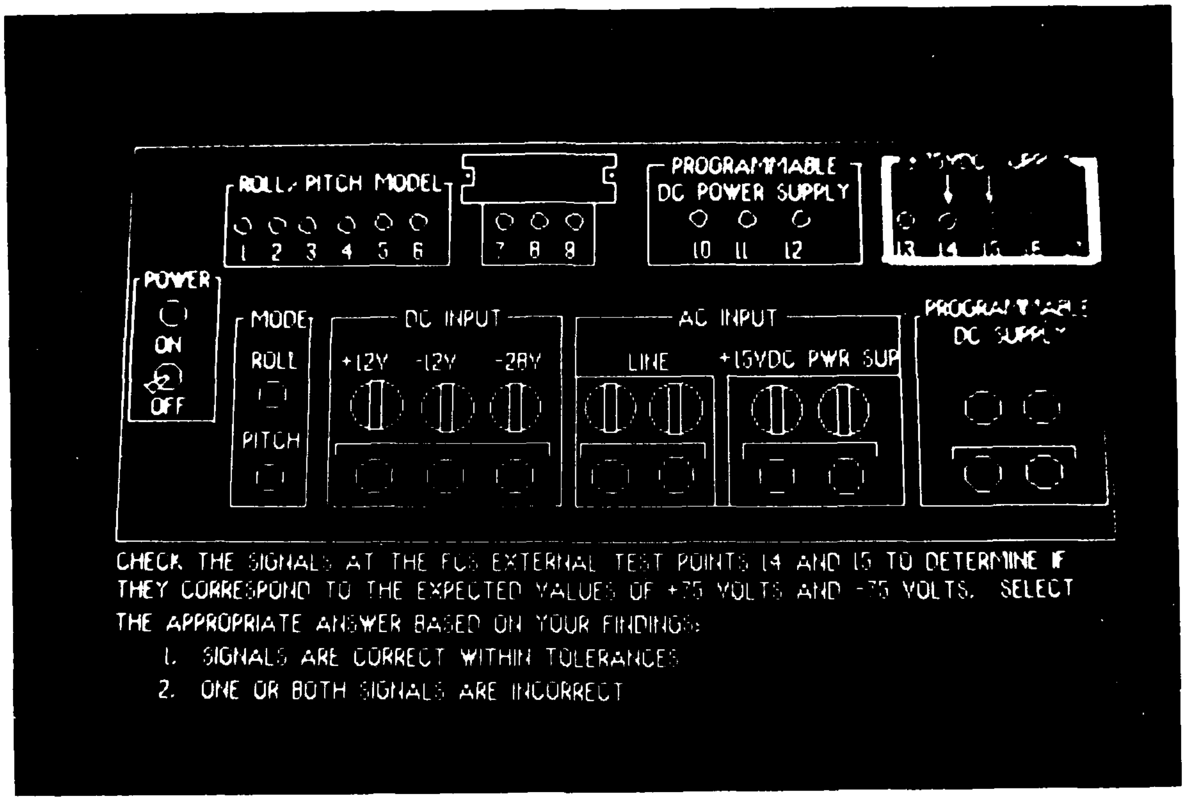
A principal goal in this effort was to demonstrate the operations of the prototype Maintainer's Associate for a variety of Air Force, industry, and research audiences. These demonstrations were designed to meet three project objectives: (a) dissemination of results, (b) validation of the prototype, and (c) formation of future R&D guidelines. The reactions of current avionics technicians were of particular interest because this information provided the basis for system validation and further development of the prototype as a job performance aid and trainer. As a complement to this perspective, the reactions of maintenance officers allowed project staff to assess the response to integrated diagnostic equipment that may be expected for future weapon and support systems. The results of these technical demonstrations are presented in this chapter.

Demonstration to Avionics Technicians

Approach

For technical personnel, a scripted demonstration was developed based on the typical troubleshooting scenario which was introduced in Chapter 3. This scenario began when a failure was indicated at test number 301982 in the automatic test sequence for the Feel and Trim LRU. By substituting a known operational Feel and Trim into the test loop and repeating the test sequence, the problem was isolated to the test station. Subsequent manual troubleshooting proceeded under the guidance of the Maintainer's Associate system. The display shown in Figure 19, for example, requested that the technician check test points 14 and 15 on the front panel of the FCS. Eventually, a malfunction in a power supply circuit of the FCS adapter was identified and appropriate repair action was indicated. This interactive sequence of Maintainer's Associate displays and keypad responses was selected to exhibit the full range of system features within the context of a fairly typical test station problem. The complete series of 18 demonstration displays required approximately 10 minutes to present. Following the scripted demonstration, technicians were encouraged to try out the Maintainer's Associate by entering one test number from a list of 60 that the system was capable of troubleshooting. Technicians would open their LRU technical orders to the test number in question and compare the Maintainer's Associate diagnostic strategy and specific requests for measurements to their own. Demonstrations were conducted with no more than three technicians per system to ensure clear visibility. Each session was prefaced by brief remarks about the purpose of the system, allowed time for questions and answers, and included a period for hands-on tryout of the Maintainer's Associate.

Two different technical groups were selected to participate in the demonstration sessions. The first consisted of six avionics instructors from the Air Training Command at Lowry AFB, Colorado. These technicians averaged more than 6 years of avionics maintenance experience, and all but one were rated at the 7-level skill classification. The second group was comprised of 10 avionics technicians from the intermediate-level F-111 maintenance shop at Cannon AFB, New Mexico, who averaged approximately 2.5 years of experience. A broad range of skilled personnel was represented in the Cannon sample, including four 3-levels, five 5-levels, and one 7-level technician.



CHECK THE SIGNALS AT THE FCS EXTERNAL TEST POINTS 14 AND 15 TO DETERMINE IF THEY CORRESPOND TO THE EXPECTED VALUES OF +75 VOLTS AND -75 VOLTS. SELECT THE APPROPRIATE ANSWER BASED ON YOUR FINDINGS:

1. SIGNALS ARE CORRECT WITHIN TOLERANCES
2. ONE OR BOTH SIGNALS ARE INCORRECT

Figure 19. Interaction Frame for Checking Test Points 14 and 15 in the FCS.

Results and Discussion

System critiques. Critique forms were self-administered by all avionics personnel who participated in a demonstration session with the Maintainer's Associate. Personnel were asked to assign ratings to each of 12 system performance factors, using a 5-point Likert scale that ranged from "very poor" (1) to "excellent" (5). Space was also provided for general comments and comments about those features they liked most and liked least. A copy of the critique form used is provided in Appendix B.

Table 2 shows the mean ratings for each performance factor as a function of each group and overall. In general, the responses were overwhelmingly positive, with both instructors and field technicians giving the system a mean overall rating of 4.6. The lowest combined mean rating concerned the range of user options (3.8) and the highest combined mean ratings were given for ease of use (4.8) and usefulness for training (4.8). Surprisingly, although the system was designed as an on-the-job troubleshooting aid, both groups rated it slightly higher on its usefulness for training than on its usefulness for job aiding. The most noticeable differences in the mean ratings of the two groups (field technicians vs. instructors) were revealed in their assessment of the display quality [3.8 vs. 5.0; t(14) = 5.28, $p < .001$], the helpfulness of explanations [4.6 vs. 3.5; t(12) = 4.16, $p < .01$], and the usefulness for job aiding [4.7 vs. 3.8; t(14) = 2.59, $p < .05$]. These latter two differences suggest that instructors may not be in the best position to judge what is perceived as helpful and useful by less experienced technicians in the field.

Table 2. Mean Ratings[a] of System Performance from Demonstration Critiques

Performance Factor	Field technicians (n = 10)	Instructors (n = 6)	Overall (n = 16)
Ease of Use	4.7	4.8	4.8
Speed of Operation	4.1	4.5	4.3
Troubleshooting Accuracy	4.4	4.4	4.4
Troubleshooting Strategy	4.7	4.5	4.6
Troubleshooting Efficiency	4.3	4.0	4.2
Range of User Options	4.0	3.3	3.8
Display Quality	3.8	5.0	4.3
Helpfulness of Explanations	4.6	3.5	4.3
Hardware Packaging	4.2	4.8	4.4
Usefulness for Job Aiding	4.7	3.8	4.4
Usefulness for Training	5.0	4.3	4.8
Overall Rating	4.6	4.6	4.6

[a]Scale: 1 = Very Poor, 5 = Excellent

Additional comments were also favorable and focused largely on the training potential of the prototype. A number of the technicians discussed the need to provide more depth to the system in terms of the malfunctions covered, explanations provided, and technical references. The features that they reportedly liked the most about the Maintainer's Associate included its simplicity and training capability, its compact size, its logical (test-loop-based) approach, and its ability to save time by avoiding the use of technical orders. Those elements liked the least included the readability of the display (too crowded) and the lack of depth in the range of problems and scope of explanations in the current prototype. Several instructors also expressed the concern that novice technicians might become dependent on the device without developing appropriate troubleshooting skills.

Formative interviews. Individual interviews were also conducted with five of the field technicians to assess in more detail their reactions to the Maintainer's Associate and their suggestions for future work. Each interview lasted approximately 20 minutes and consisted of open-ended questions about general system operation and each of the system interface features. The interview guide used to support this data collection effort is provided in Appendix B. The results of these formative interviews were generally consistent with the critique results discussed previously. Specifically, the field technicians reported that:

1. the message length and level of explanation provided by the HOW feature were appropriate, but the addition of technical orders references might be useful;

2. the graphics were quite helpful within the interaction frames, especially the test loop diagram, but the screen appeared a bit crowded at times;

3. the MARK & RETURN feature could be improved by allowing the user to begin anywhere in the troubleshooting process;

4. the WHERE-TO feature was useful but might be moreso if the information was presented in a test tree graphic format; and

5. the WHERE-FROM feature was critical for training purposes and might be improved by the addition of canned explanations and/or test tree diagrams.

Although comments were overwhelmingly favorable, additional suggestions included expanding the problem space to other TRU and even LRU malfunctions, adding a hard-copy capability, and providing more detailed information for novice technicians. Technician's comments coincided in large part with the expectations of the project staff and were typically a reflection of the established scope of the prototype effort, rather than system design limitations or operational shortcomings. Suggestions for future system development are considered more fully in Chapter 6.

Demonstration to Deputy Chief of Maintenance

Approach

DRI presented a formal briefing and scripted demonstration to the Deputy Chief of Maintenance (DCM) and members of his staff, 27th Tactical Fighter Wing, Cannon AFB. The briefing explained the background, objectives, and technical approach of the prototype. The scripted demonstration was the same one presented to avionics technicians; each demonstration took about 20 minutes followed by over an hour of discussion. The purpose of these sessions was to validate the Maintainer's Associate concept, not with respect to the accuracy of its technical detail, but with respect to its feasibility from an organizational and management perspective.

Results and Discussion

The discussion following the briefing and demonstration can be summarized in three main points. First, the Maintainer's Associate, or devices like it, were familiar to maintenance officers; and the likelihood that the Air Force would eventually employ this method of job aiding was acknowledged and accepted. The DCM took a realistic view of the Maintainer's Associate, recognizing that with any new technology, there is a need for system test, evaluation, and continual improvement.

Second, the maintenance officers were concerned with being dependent on yet another computer system. Computer-based systems have a reputation for unreliability and inaccessibility. The officers recommended that an adequate backup system be provided as part of a device deployment.

Third, this group was concerned about the potential for mental dependence on the job aid. Since the Maintainer's Associate is capable of solving troubleshooting problems, the officers were worried that avionics technicians would rely on the system and not develop or exercise their own diagnostic competency. The formal briefing outlined how this issue was explicitly addressed in system design through provision of skill multiplier features, including maintenance training simulation. The prototype was purposefully constructed not only to avoid mental dependence, but to actively support skill acquisition. It is important to note that the officers' concerns with dependence reinforced the validity of the intended purpose of the skill multiplier features--to satisfy the need for trained technical personnel.

Conclusion

Although the technical audience for the system demonstration was relatively small, technicians clearly validated the approach as implemented in the current prototype. Both classroom instructors and field personnel were favorably impressed with the performance and training potential of the Maintainer's Associate. Their suggestions for future R&D focused on realizing that potential and expanding the problem domain. The management audience was not resistant to the associate concept as the inevitable solution to existing maintenance problems with technical documentation. They were, however, concerned with how this technology will be institutionalized, especially with respect to the potential problems of physical and mental dependence.

CHAPTER 6. CONCLUSIONS AND DIRECTIONS FOR FUTURE WORK

The findings of this study suggest a number of potential areas for future development of the Maintainer's Associate system. The purpose of this chapter is, first, to summarize the accomplishments of the current work and, second, to make recommendations for future R&D in light of the anticipated near-term trends in weapon support systems. These objectives are addressed as they relate to specific topical areas of R&D in the discussion that follows.

Skill Multiplier

Next-generation weapon systems will continue the long-standing trend toward greater complexity. By virtue of this, increased diagnostic sophistication will be required just to keep even with current maintenance proficiency. This increased diagnostic sophistication, however, will not include more use of automatic test equipment in intermediate-level shops. On the contrary, the Air Force will move toward reduced dependence on the avionics intermediate shop. This may be accomplished through a combination of strategies, including: more reliable LRUs, better built-in test and associated on-aircraft diagnostic infrastructures, better integration of test strategies across maintenance levels, better data-keeping on faults, better fault isolation procedures for flight line maintenance, policies of continued system maturation beyond full-scale production, and fault-tolerant design. Human involvement in weapon system maintenance will remain, probably supported by devices like the Maintainer's Associate. Thus, the weapon system, its automatic support systems, and the human maintenance technician will all be more sophisticated diagnostically than in today's systems.

The demonstrations of the prototype Maintainer's Associate established its diagnostic competence and appropriateness for the field environment. Technicians perceived it as a good job aid and training device, the two principal features of a skill multiplier, because it enables novice technicians to solve diagnostic problems beyond their own level of competence and teaches them how to solve future problems on their own.

One of the ways to promote skills acquistion is through problem-solving exercises. For maintenance technicians, these problem-solving exercises generally take the form of troubleshooting simulations. Given the fact that the Maintainer's Associate has diagnostic competency in the expert system, and given that it is possible to access the reasoning or inference mechanism behind this competency, it provides the basis for construction of a troubleshooting coach. In its basic conception (refer to Chapters 1 and 2), this coach is simply an inversion of the expert system. The technician poses the questions as to where and what to test, and the expert system answers with the (simulated) findings. The coach forces the student to anticipate each of its own processing steps in advance; and, if the student's next step differs from the coach's, it intervenes immediately with corrective action. The inversion, however, can, and should, go deeper than this.

By designing the coach to the extent that the technician is forced to emulate the actions of the expert system's inference procedure, the technician learns a successful general diagnostic strategy (i.e., the inference cycle of establish-refine) and the problem-specific details (i.e., the rule base) on which it operates.

Future work should focus on implementing this coach and coupling it with a curriculum-sequencing module and a student model to yield a complete intelligent tutorial system. The curriculum-sequencing module would be able to determine what simulated fault (and what part of the troubleshooting task for that fault) out of all possible faults would best serve to advance the skill level of the technician as measured against on-the-job training objectives. The student model would be a detailed record of the technician's successes and failures with each simulation exercise.

A key outcome of this work would be demonstrating that the same knowledge base useful in job aiding is useful in training. Were this true, it would pave the way to the successful integration of job aiding and training and present a revolutionary new way to develop the skills in a cadre of technicians. More generally, the successful development and evaluation of such a troubleshooting coach would supplement what is known about building intelligent tutoring systems.

Skill Integrator

The design goal for future weapon systems will not only change, but the process by which this goal is achieved will be new. First, fault detection and fault isolation concerns will be addressed as a system and the various diagnostic technologies--BIT, ATE, job aids, MIS--will not be developed independently, but as an integrated whole. Computer-aided design, engineering, and manufacturing have already had, and will continue to have, a large impact on the design process. This computer-aided support capability establishes a closed information loop from design to support, and back again, throughout which the device model serves as the basis of communication, coordination, and integration among the phases of a weapon system. This closed-loop approach reinforces the idea that system maturation never ends. Because of their important role in serving as a source of information on weapon and support system performance, trained technical personnel will always participate in maintenance. The skill integrator interface of the Maintainer's Associate was designed to support this role.

The skill integrator works cooperatively with skilled technicians to solve problems and capture the human technician's diagnostic insights as they arise. The foundation for this feature is the extensible, modularized rule base of the Rule-Kit expert system. The expert system architecture is capable of interpreting, in a uniform representation, rules of inference derived from a description of system structure, as well as informal, experiential heuristics. Although this basis for adding human diagnostic insights to the rule base exists, a suitable debriefing interface was not implemented in the Maintainer's Associate prototype.

Future work to be done on this debriefing interface offers a variety of options. Several events could initiate the debriefing, ranging from user initiative to the system's failure to isolate the fault. In either case, the architecture and knowledge base of the expert system could provide context, semantics, and even syntax in which to carry on a discourse with the user regarding a new diagnostic rule. The problems of natural language are greatly simplified when the context is constrained and when semantics and syntax are specified. The existing architecture and rule base makes the notion of talking to a computer about an insight a far more likely possibility. Successful implementation of this debriefing interface would yield results of general interest to the field of natural language processing.

A second aspect of the maintainer's associate skill integrator interface is cooperative human-computer problem-solving. Very little basic theoretical work has been conducted in this area. The problem-solving strengths of humans and computers have been contrasted in a descriptive way, but no prescriptive design for combining these has been approached. The browse, jump-ahead, and debriefing features of the Maintainer's Associate are only a beginning in what may become an increasingly important field. Basic R&D in this area could be focused by limiting problem-solving to troubleshooting in the electronics equipment domain.

It is particularly important that an associate be able to work with both expert and novice technicians in appropriately different ways. The novice may rely on the system for all diagnostic reasoning and data-gathering. The expert needs assistance that is flexible, because observations and preliminary reasoning have probably been done independently.

The MARK & RETURN feature on the Maintainer's Associate implemented a partial solution to this concern. This feature allows the technician to explore where the system would go, diagnostically, given different answers to its questions. The technician, however, is still unable to make assertions independent of whether the assertion is an acceptable answer to the question the Maintainer's Associate is currently posing. Ideally, technicians should be able to take the initiative by making such assertions, both of evidence observed and of probable causes suspected. These assertions should also be able to be made outside the context of the expert system's current state (i.e., they should not have to match the assertions the expert system is currently processing); and once the technician has entered these assertions, they should impact the flow of inferences being made.

The above features represent only one of many possible design strategies that facilitate cooperative human-computer problem-solving. It is an approach that mitigates against the inevitable frustration experts would feel in dealing with a system designed for novices. The Maintainer's Associate expert system architecture provides a foundation for further development of the mixed-initiative approach to troubleshooting.

One aspect of the maintainer's associate concept that was not explored in this effort was the ability to prebrief technicians. A prebriefing interface would integrate corporate-wide skills by providing technicians access to the MIS records for all past repairs of the system under test. The development of MIS systems has been accomplished for many specific weapon systems. In the future, these systems should capitalize on a design to close the information loop based on the device model, as discussed in Chapter 1. MIS systems would also benefit from the notion of episodic memory and the use of AI techniques in this area (see Kolodner, 1983) to develop the database of maintenance events. This could be useful in characterizing and categorizing maintenance events; in recognizing new, novel, or aberrant events; and in facilitating the return flow of information back from support activities to design activities, where rectifying design changes may be made.

For the prebriefing interface, future work should also be directed at rounding out the complement of four information resources which the Maintainer's Associate incorporates. This would include the incorporation of an interactive gateway to the weapon system's maintenance information system and to its supporting technical documentation, principally schematics, illustrated parts breakdowns, and removal, installation, calibration, alignment, safety, and other information.

Knowledge Acquisition

Hybrid Diagnosis

Demonstrating an efficient knowledge acquisition strategy capable of integrating both specification- and symptom-based knowledge was a project objective that was achieved largely by representing all rules in the establish-refine formalism. Although no automated tools were used to develop specification-based diagnostic rules, the knowledge engineers, in conjunction with the SMEs, used a specification-based process. This process analyzed the structure (topology, dependency) of the system under test, derived a diagnostic strategy (test tree) from this, and converted the strategy to rules in Rule-Kit's syntax. Although this process was conducted using the technician's mental model, its basis was system structure, not empirical symptom-fault associations. (This process and its results are described in more detail in Chapter 4.)

It must be recognized that however well a diagnostic system may be operationalized using only specification-based knowledge, some faults will go undetected because of incompleteness or simplification in the device model. Symptom-based rules fill the gap in the knowledge base when the manifest symptom-failure associations for previously undiagnosable faults are determined. This project showed that a hybrid diagnostic approach offers a feasible method for resolving this shortcoming of specification-based diagnostics. The project staff found, however, that for this diagnostic task, the great preponderance of rules were specification- rather than symptom-based. Knowledge engineers initiated their work consciously seeking out each type of inference, but in only a very few cases was symptom-based knowledge employed. When found, however, it was possible to use symptom-based knowledge together with the specification-based knowledge in a uniform representation.

Reconfigurable Systems

A major contribution of this project was the implementation of an AI approach to troubleshooting reconfigurable systems. The 6883 test station is really 400 different systems, depending on the LRU test number. The challenge posed was how to troubleshoot a system of multiple states without developing multiple sets of rules. The solution was to conceive of the system at a level of abstraction at which the system's description remained invariant across states. In the case of automatic test stations, this abstraction was the test loop. (Other reconfigurable systems will have other invariant abstractions.) The system was abstractly conceived, and the diagnostic strategy was applied to the components of the abstraction. This was an instance of the hierarchical decomposition approach to diagnosis, where the hierarchical abstraction not only removed details of connection but also of state.

The diagnostic strategy derived from the abstract system model identified where to test (in terms of topology or dependency), but not where to test physically. Although abstracted, the system itself remains a concrete physical object. Thus, there was the problem of how to make the diagnostic strategy's request to check at the input to an abstract region match the actual corresponding physical location and expected signal value. This translation was accomplished through use of a parser, which compiled lists of physical locations and expected measurement values by abstract region for each system state. The parser's principal source of knowledge about system state was the UUT's test program set; indeed, the test program set is what defines the test station's state. The diagnostic test strategy was written independently of state, and the instantiation of the abstraction into a physical location and expected value was accomplished by the parser.

Authoring System

There are three types of tools that can be of particular use to the knowle 'ge engineer: (a) a rule base editor that operates in the context of a diagnostic consultation, (b) a verification routine that exercises all possible paths through the rule base and reports inconsistencies, and (c) a system-specification-to-diagnostic-strategy converter.

One objective of this work was to demonstrate that knowledge base development could be accomplished by nonprogrammers. This was achieved at the outset, simply due to the nature of the expert system architecture selected. In Rule-Kit, control and data are separated. A uniform data format is interpreted in a uniform way by the expert system's inference engine. Once a knowledge engineer has interpreted a diagnostic strategy in the context of the expert system's architecture and rule format, it is easy to develop the knowledge base. This project made the task of the knowledge engineer even easier by providing the ability to enter or edit rules in the context of their execution with the glass box editor. Thus, the glass box editor enables nonprogrammers to input and manipulate the knowledge base. This editor may be a good starting point for further work on the skill integrator interface. The graphics work station and the validator-verifier were other system development tools which helped complete and debug the knowledge base. Many routine procedural errors were detected by the validator-verifier and then removed.

In this particular project, an aid for converting descriptions of structure into diagnostic strategies was not developed. As reviewed in Chapter I, many such tools exist. Since the knowledge engineers employed largely a specification-based approach, it is concluded that this stage of knowledge base development could be supported with a computer-based tool. The tool should permit flexible interaction with the knowledge engineer, both in inputting the device model and in interactively editing the tool's output.

Institutionalization

Issues surrounding the organizational impact of a new technology must be addressed far in advance of efforts to use the technology. Before advanced development activities with the Maintainer's Associate can be undertaken, issues surrounding its institutionalization need to be explored. Two institutionalization issues that were raised during the demonstrations at the end of this project need further examination: reliability and acceptance. How can the maintainer's associate system be made sufficiently reliable so that it performs reliably in an operational environment? Once this is adequately answered, how can users be persuaded and convinced of this reliability so that they accept this new approach to technical job aiding and training; and what other organizational factors will enhance its introduction, acceptability, and utility?

Advanced development efforts with associate systems will also require addressing systems engineering issues. The overall maintainer's associate system, and its development and operation, would need to be explored in detail. Ideally, the system would comprise a worldwide information network, with links from the design engineers to the flight line. Important questions immediately arise with respect to this scenario: How will configuration control be maintained? How and how often will updates be managed? How will this information, tantamount to the health, well-being, and weakness of our weapon systems, be secured? What will happen if the system's satellites are down? How will the system work if a unit is deployed to a remote area?

Systems engineering issues also extend to the classroom. The integrated job aiding and training a maintainer's associate makes possible will undoubtedly have an impact on the Air Force's training establishment, the Air Training Command. Specifics of this impact need to be identified and varieties of responses need to be analyzed so that changes can be made in anticipation of, and not in reaction to, the changing technology that will bring the maintainer's associate to fruition.

Due to the success in developing the protoype, its successful demonstration, and the maintenance concerns and scenario of the future, it is recommended that advanced development versions of the Maintainer's Associate be developed and field tested as soon as is practicable. This recommendation is consistent with the finding of the National Academy of Sciences Committee on Fault Isolation in Air Force Weapon and Support Systems. The prototype, without any of the enhancements possible with exploratory development, if scaled up to a realistic scope, could prove to be a useful and acceptable advance in weapon systems support. The deployment of such a maintainer's associate system for an Air Force weapon system is an achievable near-term goal. In preparation for this goal, serious studies of the benefits, costs, and risks of the maintainer's associate need to be undertaken. As a first step, this project has helped define the concept, reduce the risk involved, and identify the benefits that might accrue.

REFERENCES

Air Force Human Resources Laboratory. (1984, June). Artificial intelligence in maintenance: Proceedings of the Joint Services Workshop (AFHRL-TR-84-25, AD-A145 349). Lowry AFB, CO: Training Systems Division, Air Force Human Resources Laboratory.

Anderson, J. R., Boyle, C. F., Farrell, R., & Reiser, B. (1984). Cognitive principles in the design of computer tutors. In Proceedings of the Sixth Annual Conference of the Cognitive Science Society. Boulder, CO: Institute of Cognitive Science and the University of Colorado.

Andre, W. L., & Wong, J. T. (1975). Mathematical structure and optimal diagnostic technique for maintenance analysis. In Proceedings of the 36th Military Operations Research Symposium. Moffett Field, CA: U.S. Army Air Mobility R&D Laboratory, Ames Research Center.

Bobrow, D. G., & Hayes, P. J. (Eds.). (1984, December). Qualitative reasoning about physical systems (Special volume). Artificial Intelligence, 24(1-3).

Bonissone, P. P., & Johnson, H. E., Jr. (1984, June). DELTA: An expert system for diesel electric locomotive repair. In Artificial intelligence in maintenance: Proceedings of the Joint Services Workshop (AFHRL-TR-84-25, AD-A145 349). Lowry AFB, CO: Training Systems Division, Air Force Human Resources Laboratory.

Brown, A. (1977). Qualitative knowledge, causal reasoning, and localization of failures (MIT AI-TR-3620). Cambridge, MA: Massachusetts Institute of Technology.

Brown, A., & Sussman, G. J. (1974). Localization of failures in radio circuits--a study in causal and teleological reasoning (MIT AI Memo No. 319). Cambridge, MA: Massachusetts Institute of Technology.

Brown, J. S., Burton, R., & de Kleer, J. (1982). Pedagogical, natural language and knowledge engineering techniques in SOPHIE I, II and III. In D. Sleeman & J. S. Brown (Eds.), Intelligent tutoring systems. New York: Academic Press.

Cantone, R. (1984). Model-based probabilistic reasoning for electronics troubleshooting. In Artifical intelligence in maintenance: Proceedings of the Joint Services Workshop (AFHRL-TR-84-25, AD-A145 349). Lowry AFB, CO: Training Systems Division, Air Force Human Resources Laboratory.

Cantone, R., Lander, B. W., Marrone, M. P., & Gaynor M. W. (1984). IN-ATE/2: Interpretating high-level fault modes. In Proceedings of the First Conference on Artificial Intelligence Applications. Denver, CO: IEEE Computer Society.

Cantone, R., Pipitone, F. J., Lander, W. B., & Marrone, M. P. (1983). Model-based probabilistic reasoning for electronics troubleshooting. In A. Bundy (Ed.), Proceedings of the Eight International Joint Conference on Artifical Intelligence (IJCAI-83). Los Altos, CA: William Kaufmann.

Chandrasekaran, B. (1983). Towards a taxonomy of problem solving types. The AI Magazine, 4(1), 9-17.

Clancey, W. J. (1984). Classification problem solving. In Proceedings of the National Conference on Artificial Intelligence, American Association for Artifical Intelligence, Austin, TX. Los Altos, CA: William Kaufmann.

Cramer, M L., et al. (1982). Logic model analysis and standard maintenance information display system (SMIDS) (USAAVRADCOM-TR-81-D-45). Fort Eustis, VA: U.S. Army Research and Technology Laboratories, Applied Technology Laboratory.

Davis, R. (1983). Diagnosing via causal reasoning: Paths of interaction and the locality principle. In Proceedings of the National Conference on Artificial Intelligence, American Association for Artifical Intelligence, Washington, DC. Los Altos, CA: William Kaufmann.

Davis, R. (1984). Diagnostic reasoning based on structure and behavior. Artificial Intelligence, 24(1-3), 347-410.

Davis, R., Shrobe, H., Hamscher, W. Wieckert, K., Shirley, M., & Polit, S. (1982). Diagnosis based on description of structure and function. In Proceedings of the National Conference on Artificial Intelligence, American Association for Artifical Intelligence, Pittsburgh, PA. Los Altos, CA: William Kaufmann.

Davison, J. (1984). Expert systems in maintenance diagnostics for self-repair of digital flight control systems. In Artificial intelligence in maintenance: Proceedings of the Joint Services Workshop (AFHRL-TR-84-25, AD-A145 349). Lowry AFB, CO: Training Systems Division, Air Force Human Resources Laboratory.

de Kleer, J. (1976). Local methods for localizing faults in electronic circuits (MIT AI AIM-394). Cambridge: Massachusetts Institute of Technology.

de Kleer, J. (1984). AI approaches to troubleshooting. In Artificial intelligence in maintenance: Proceedings of the Joint Services Workshop (AFHRL-TR-84-25, AD-A145 349). Lowry AFB, CO: Training Systems Division, Air Force Human Resources Laboratory.

DETEX Systems, Inc. (n.d.). What is LOGMOD? and how it helps you! Villa Park, CA: DETEX Systems, Inc.

Fink, P. K., Lusth, J. C., & Duran, J. W. (1984). A general expert system design for diagnostic problem solving. In Proceedings of the IEEE Workshop on Principles of Knowledge-Based Systems, Institute of Electrical and Electronics Engineers Computer Society, Denver, CO. Silver Spring, MD: IEEE Computer Society Press.

General Dynamics Electronics Division. (1984). Rule-Kit: A set of tools for building rule-based diagnostic systems. San Diego, CA: Author.

Genesereth, M. R. (1982). Diagnosis using hierarchical design models. In Proceedings of the National Conference on Artificial Intelligence, American Association for Artificial Intelligence, Pittsburgh, PA. Los Altos, CA: William Kaufmann.

Genesereth, M. R. (1984). The use of design descriptions in automated diagnosis. Artificial Intelligence, 24(1-3), 411-436.

Hamscher, W., & Davis, R. (1984). Diagnosing circuits with state: An inherently underconstrained problem. In Proceedings of the National Conference on Artifical Intelligence, American Association for Artificial Intelligence, Austin, TX. Los Altos, CA: William Kaufmann.

Hinchman, J. H., & Morgan, M. C. (1984). Application of artificial intelligence to equipment maintenance. In Artificial intelligence in maintenance: Proceedings of the Joint Services Workshop (AFHRL-TR-84-25, AD-A145 349). Lowry AFB, CO: Training Systems Division, Air Force Human Resources Laboratory.

Hollan, J., Stevens, A., & Williams, M. (1980). STEAMER: An advanced computer-assisted instruction system for propulsion engineering. Paper presented at Summer Simulation Conference, Seattle, WA.

Johnson, R. C. (1981). Integrated maintenance information system: An imaginary preview (AFHRL-TP-81-18, AD-A104 224). Wright-Patterson AFB, OH: Logistics and Technical Training Division, Air Force Human Resources Laboratory.

Keller, R. A. (1985, November). Human troubleshooting in electronics: Implications for intelligent maintenance aids (AFHRL-TP-85-34, AD-A161 832). Lowry AFB, CO: Training Systems Division, Air Force Human Resources Laboratory.

King, J. J. (1982). Artificial intelligence techniques for device troubleshooting (Computer Science Laboratory Technical Note Series CSL-82-9). Palo Alto, CA: Hewlett-Packard.

Kolodner, J. L. (1983, October-December). Maintaining organization in a dynamic long-term memory. Cognitive Science, 7(4), 243-280.

Laffey, T. J., Perkins, W. A., & Nguyen, T. A. (1984). Reasoning about fault diagnosis with LES. In Proceedings of the First Conference on Artificial Intelligence Applications. Denver, CO: IEEE Computer Society.

Lahore, H. (1984). Artificial intelligence applications to testability (RADC-TR-84-203). Griffis AFB, NY: Rome Air Development Center, Air Force Systems Command.

Longendorfer, B. (1981). Computer-aided testability analysis of analog circuitry. In AUTOTESTCON '81, Institute of Electrical and Electronics Engineers.

McDermott, D. V. (1976). Flexibility and efficiency in a computer program for designing circuits. Cambridge, MA: Massachusetts Institute of Technology.

McDermott, D. V., & Brooks, R. (1982). ARBY: Diagnosis with shallow causal models. In Proceedings of the National Conference on Artificial Intelligence, American Association for Artificial Intelligence, Pittsburgh, PA. Los Altos, CA: William Kaufmann.

National Academy Press. (in press). Isolation of faults in Air Force weapon and support systems. Washington, DC: Author.

National Security Industrial Association, Automatic Test Committee, Logistics Management Committee. (1983). Proceedings of the Conference on Integrated Diagnostics. Tampa, FL: Author.

National Security Industrial Association, Automatic Test Committee, Logistics Management Committee. (1984a). Proceedings of the Symposium on R&D for Weapon Support and Logistics. Alexandria, VA: Author.

National Security Industrial Association, Automatic Test Committee, Logistics Management Committee. (1984b). Report for the Department of Defense on the implementation of integrated diagnostics. Washington, DC: Author.

Pipitone, F. (1984). An expert system for electronics troubleshooting based on function and connectivity. In Proceedings of the First Conference on Artificial Intelligence Applications. Denver, CO: IEEE Computer Society.

Richardson, J. J., Keller, R. A., Maxion, R. A., Polson, P. G., & DeJong, K. A. (1985, October). Artificial intelligence in maintenance: Synthesis of technical issues (AFHRL-TR-85-7, AD-A160 863). Lowry AFB, CO: Training Systems Division, Air Force Human Resources Laboratory.

Rouse, W. B. (1984). Models of natural intelligence in fault diagnosis tasks: Implications for training and aiding of maintenance personnel. In Artificial intelligence in maintenance: Proceedings of the Joint Services Workshop (AFHRL-TR-84-25, AD-A145 349). Lowry AFB, CO: Training Systems Division, Air Force Human Resources Laboratory.

Simpson, W. R., & Agre, J. R. (1983). Adaptive fault isolation with learning. In AUTOTESTCON '83 (83CH1968-7), Institute of Electrical and Electronics Engineers, Fort Worth, TX. New York: IEEE.

Simpson, W. R., & Balaban, H. S. (1982). The ARINC research system testability and maintenance program (STAMP). In AUTOTESTCON '82. New York: IEEE.

Stallman, R. M., & Sussman, G. J. (1977). Forward reasoning and dependency-directed backtracking in a system for computer-aided circuit analysis. Artificial Intelligence, 9, 135-196.

Sussman, G. J., & Steele, G. L., Jr. (1980). CONSTRAINTS--A language for expressing almost-hierarchical descriptions. Artificial Intelligence, 14, 1-39.

Tanner, M. C., & Bylander, T. (1984). Application of the CSRL language to the design of expert diagnosis systems: The Auto-Mech experience. In Artificial Intelligence in Maintenance: Proceedings of the Joint Services Workshop (AFHRL-TR-84-25, AD-A145 349). Lowry AFB, CO: Training Systems Division, Air Force Human Resources Laboratory.

Williams, T., & Hinchman, J. (1983). A knowledge-based maintenance aid for diagnosing faults in electronic equipment. In AUTOTESTCON '83 (83CH1968-7), Institute of Electrical and Electronics Engineers, Fort Worth, TX. New York: IEEE.

Wong, J. T., & Andre, W. L. (1976). Some generic properties of a logic model for analyzing hardware maintenance and design concepts. In Proceedings of the Symposium on Applications of Decision Theory to Problems of Diagnosis and Repair. Moffett Field, CA: U.S. Army Air Mobility R&D Laboratory, Ames Research Center.

Wong, J. T., & Andre, W. L. (1981). Logic analysis of complex systems by characterizing failure phenomena to achieve diagnosis and fault-isolation (NASA Tech. Memo 81291). Moffett Field, CA: National Aeronautics and Space Administration.

APPENDIX A: TECHNICAL DATA FOR PARSER DEVELOPMENT

Technical orders manual	Figure
33A1-378-2	8-2
33A1-10-112-2	8-1
33D3-9-99-2	8-1
33D3-9-100-2	8-2
33D3-9-101-2	8-2
33D3-9-102-2	6-1 8-2
33D3-9-103-2	6-1 8-2 8-6
33D7-15-2	9-1 9-10 9-11 9-14 9-15 9-16 9-18
33D7-42-1-132	4-1 4-4 7-14 7-15 7-16 9-245
33DA8-21-2	8-2
5A9-2-42-28-1	2-10 2-11

APPENDIX B: DATA COLLECTION INSTRUMENTS

CRITIQUE

Please take a few minutes to record your impressions of the prototype Maintainer's Associate (MA) that has just been demonstrated. Your comments are vital for evaluating the success of the project to date as well as for determining the direction this research might take in the future. Thank you!

What is your current skill level? ______________________________

How many years experience do you have in electronics maintenance? __________

How many years experience do you have with 6883/73 test equipment? __________

Did you actually use the MA? __________ or just observe the MA? __________

How would you rate the performance of the MA system on the following factors:

	Very Poor				Excellent
Ease of use?	1	2	3	4	5
Speed of operation?	1	2	3	4	5
Troubleshooting accuracy?	1	2	3	4	5
Troubleshooting strategy?	1	2	3	4	5
Troubleshooting efficiency?	1	2	3	4	5
Range of user options?	1	2	3	4	5
Display quality?	1	2	3	4	5
Helpfulness of explanations?	1	2	3	4	5
Hardware packaging?	1	2	3	4	5
Usefulness for job aiding?	1	2	3	4	5
Usefulness for training?	1	2	3	4	5
Overall rating of performance?	1	2	3	4	5

What did you like most about the MA system? ______________________________

____

What did you like least about the MA system? ______________________________

____

Any other comments? ____

____

INTERVIEW GUIDE

Date: ______________

Time: ______________

Location: __________________________

Subject: ___________________________

Interviewer: _______________________

1. Read the following instructions:

   "The purpose of this tryout session is to get your honest reactions to the various features of the Maintainer's Associate. This is a prototype system, and your comments are important for future development efforts. For the first part of this session, I would like to guide you through a possible troubleshooting situation, collecting your reactions (if any) at each step in the process. During the second part of the session, you will have an opportunity to try out the MA system entirely on your own."

2. Begin the demonstration part with the SME entering all choices. Try to get comments on the following features as they seem appropriate.

   a. HOW

      Appropriate level of explanation for 3- and 5-level technicians?

      Message length OK?

      Need additional details, references, or diagrams?

   b. INTERACTION FRAMES

      Are directions clear?

Does the system have the right 'associate' tone, or too authoritative?

Are the graphics useful?

Are the alternatives clear and appropriate?

c. MARK & RETURN

Is this feature useful?

Is it easy to use? What would make it better?

d. WHERE TO

Does this feature give you enough explanation?

Is it useful?

What might make it better? Test loop diagrams? Binary test tree?

e. WHERE FROM

Is this feature important/useful?

Is the information presented clearly?

What might make it better?

3. Allow the technician to try out the system for a short time alone. Record any comments or problems.

4. Question the technician about additional features and future plans for the system (e.g., possible notes file, simulation capability, canned why's).

5. Question the technician about hardware features.

   a. Speed of response

   b. Overall size

   c. Display quality

   d. Input mode

6. Have the technician complete the standard CRITIQUE and attach to this form.

U.S. GOVERNMENT PRINTING OFFICE:1986-[illegible]

END
2-87
DTIC